RSX-11/IAS SIG DECUS U.S. CHAPTER

THE MULTI-TASKER

Volume 15, Number 6 December/January 1982

The Newsletter of the RSX-11/IAS Special Interest Group

Contributions should be sent to: Editor, The Multi-Tasker, c/o DECUS, One Iron Way, MR2-3/E55, Marlboro, MA 01752

European members should send contributions to: Colin A. Mercer, Tennant Post, High Street, FAREHAM, PO16 7BQ, Hants, England

Members in Australia or New Zealand should send contributions to: Clive Edington, CSIRO, Computing Research 314 Albert St., East Melbourne, VIC 3002, Australia

Letters and articles for publication are requested from members of the SIG. They may include helpful hints, inquiries to other users, reports on SIG business, summaries of SPR's submitted to Digital or other information for the members of RSX-11/IAS SIG.

All contributions should be "camera-ready copy" e.g. sharp black type in a 160x240 mm area (8 1/2" x 11" paper with 1" margins) and should not include xerox copies. If you use RUNOFF to prepare your contribution the following parameters have been found to be satisfactory:

.PAPER SIZE 60,80 .LEFT MARGIN 8 .RIGHT MARGIN 72 .SPACING 1

These parameters assume output on a lineprinter with a pitch of 10 char/inch. Adjust the parameters to maintain the same margins if another pitch is used.

TABLE OF CONTENTS

Columns

Articles

Special Section

READ THIS FIRST

DIGITAL is retracting a patch published in the January, 1982 RSX-11M/S RSX-11M-PLUS Software Dispatch. The patch is sequence 2.1.1.x and patches the executive module DRREG for errors in the attach/detach region directives. Do not apply this patch because it has unexpected side effects. A correction will be published in the future.

The RSX/IAS SIG Executive Committee election is over. The "Chairman's Corner" by new SIG chairman, George Hamma, lists the current members of the Executive and Steering Committees. This is followed by the names and addresses of all members.

Please read the editorial on Software Maintenance Services and respond as soon as possible to the questionnaire at the end of this issue. This is especially important for sites who are happy with the services or do not use them so the survey results accurately reflect the feelings of the entire user community. Experience has shown that less than 10% of the Multi-Tasker mailing list responds to surveys. Unfortunately, this particularly survey needs close to 100% response in order to be effective. **PLEASE ANSWER THE QUESTIONNAIRE NOW!**

FROM THE EDITOR

This is the combined December 1981, January 1981 issue. It has some of the highlights from the Los Angeles Symposium: brief notes from the Digital product panels and the transcript of the IAS Question and Answer Session. Next month will have even more symposium articles, including the RSX-11M Question and Answer Session transcript. Also, this issue has the last part of the RSX/IAS SIG Tape Abstracts. The RUNOFF index will not be printed because it is being revised. As soon as I get the document index in good order, it will be submitted to the DECUS library along with the master tape directories.

There are many good articles and even more on my desk waiting for publication. If you have made a submission and it is not in this issue, be patient, it will be in February's. One of the best things that happened to me at Los Angeles was scheduling an impromptu meeting for Multi-Tasker volunteers and having over 100 people show up. It will take some time to get organize, but we should be up to speed by the summer.

There are a couple of minor changes starting with this issue. This issue and all future editions will be page numbered starting with one (1). No longer will all pages in a volume be numbered sequentially. Also, volume numbers will now run through the DECUS fiscal year - July to June. This is Volume 15, Number 6 and not Volume 16, Number 1 as it would have been in the old scheme.

Keep the submissions coming in. Now that there is a true Multi-Tasker staff, we can take anything you can dish out! The following are the remaining publication deadlines for the year.

	February	(Vol. 15, No. 7)	Friday, 29 January
*	March	(Vol. 15, No. 8)	Friday, 19 February
	April	(Vol. 15, No. 9)	Friday, 26 March
	May	(Vol. 15, No. 10)	Friday, 23 April
	June	(Vol. 15, No. 11)	Friday, 21 May
	July	(Vol. 16, No. 1)	Friday, 25 June
	August	(Vol. 16, No. 2)	Friday, 23 July
	September	(Vol. 16, No. 3)	Friday, 20 August
*	October	(Vol. 16, No. 4)	Friday, 17 September
	November	(Vol. 16, No. 5)	Friday, 22 October
	December	(Vol. 16, No. 6)	Friday, 19 November

* Last issue which will which readers before symposium.

I would like to extend my personal thanks to Ray French for his leadership and support over the last three years as RSX/IAS SIG chairman. The Los Angeles symposium was filled with enthusiasm and excitement, due in large part to Ray. Fortunately, we will not miss Ray, because he has merely moved over to planning coordinator position in the SIG. Unfortunately, we will have to share him with all of DECUS because of Ray's other new position as DECUS U.S. Chapter SIG Coordinator.

Ralph Stamerjohn
Multi-Tasker Editor

Phone: (314) 694-4252 (3-5 pm, CST)

CHAIRMAN'S CORNER

The RSX-11/IAS SIG Executive Committee election results were announced at the Fall 1981 DECUS Symposium by Ray French, outgoing SIG Chairman. Ray now holds the SIG Coordinator post, acting on behalf of all the SIGs for the DECUS Executive Board. The elected officers and their resulting roles are as follow (see the following article for addresses and phone numbers):

George Hamma	RSX/IAS SIG Chairman
Phil Cannon	Software Coordinator
Legare Coleman	Menu Coordinator
Ray French	Planning Coordinator
Marge Knox	Projects Coordinator
Jim Neeland	Tape Copy Coordinator
Ralph Stamerjohn	Multi-Tasker Editor

In addition, other Steering Committee positions were also filled:

Gregg Church	ALUG - East
Bob Denny	ALUG - West
Glen Everhart	DECUS Library Representative
Mike Fraser	User Group Coordinator
Bob Hassinger	Publications Coordinator
Marge Knox	Handouts Coordinator
Jim McGlinchey	Symposia Coordinator

Bill Tippie	Papers Coordinator
Tom Viana	Pre-Symposia Seminars
Tom Viana	Working Group Coordinator

At the Symposium, three new Working Groups were also formed, bringing the total up to eleven. The following working group chairman are also members of the Steering Committee:

Bill Burton	RSX-11M Unsupported Versions
Bill Carroll	Structured Fortran
Jim Downward	System Performance and Accounting
Glen Everhart	DECUS Library
Mike Fraser	Virtual Disks
John Jenkinson	IAS
Mark King	Process Control
Terry Medlin	File System and I/O Drivers
Jim Neeland	SIG Tape Collection
Bob Turkelson	SRD
Tom Viana	Training

Much confusion and many problems resulted from the cancellation of some of the announced Pre-Symposia Seminars. The cancellations resulted from a lack of advance subscription for the Seminars at the time an unpublished deadline arrived for the DECUS staff. Cost estimates had been made for the Seminars, and a minimum number of attendees had been set for each in advance. Those seminars not achieving their minimum signups were cancelled.

The SIG Steering Committee resolved to present all future Pre-Symposia Seminars offered, without qualification by minimum signup. To not do so is unfair to the presenters, who still have to make their advance materials and should not face a subsequent cancellation. Attendees should also be able to count on meetings happening, both as reasonable expectations and in support of always difficult travel arrangements.

The Symposium activities included 30 mini-tutorial sessions, each packed with technical information on a specific area of interest. Those were generally deemed a success and will be repeated at the next Symposium (in form). There were problems with coordinating sessions between the two hotels and some scheduling problems provided rather entertaining sprints over the intervening hill by our distinguished speakers. As is usually the case, the sessions for the next Symposium (Spring 1982 in Atlanta) were selected before the completion of the meetings in Los Angeles. The Atlanta site will be a single location, and the problems unique to the Los Angeles arrangements are not expected to repeat.

The SIG activities for the next fiscal year (starting July 1982) were outlined as the funding-allocation process continues within DECUS. Several SIG members contacted Steering Committee members at the Symposium, expressing an interest in participating further in SIG activities. Those inquiries were appreciated, and other interested parties are invited to contact any of the SIG Steering Committee members or the SIG Chairman via the DECUS office.

As a personal note, I found that the five-day format was a success. The measure I used was the almost complete lack of time slots over five days where there was no session I was interested in attending. The combination between the new format and a modification to the scheduling process eliminated most conflicts

between sessions and placed them very accessible. That made the break arranged in the evening sessions even more appreciated, as the intensity of activity was, if anything, even higher than usual. For me, this was the most productive Symposium for many years.

George Hamma
RSX/IAS SIG Chairman

RSX/IAS SIG STEERING COMMITTEE

* Indicates Executive Committee position

ADMINISTRATION

SIG CHAIRMAN
* George Hamma

Synergistics Technology, Inc.
20065 Stevens Creek Blvd.
Cupertino, CA 95014
Phone: (408) 253-5800

DECUS REPRESENTATIVE
* Paula Morin

DECUS
One Iron Way, MR2-3/E55
Marlboro, MA 01752
Phone: (617) 467-4141

DIGITAL REPRESENTATIVE
* Steve Paavola (RSX-11M/M+)

Digital Equipment Corporation
110 Spit Brook Road, ZK1-3/J33
Nashua, NH 03061

DIGITAL REPRESENTATIVE
Cliff Harvey (IAS)

Digital Equipment Corporation
MK1-2/A8
Merrimack, NH 03054

PLANNING

PLANNING COORDINATOR
* Raymond B. French

Boeing Commercial Airplane Company
P.O. Box 3707 MS: 9W-31
Seattle, WA 98124
Phone: (206) 237-6192

SIG PLANNING
Raymond B. French

(same address)

DIGITAL PRODUCTS
Unassigned

SYMPOSIA

SYMPOSIA COORDINATOR
James McGlinchey

P.O. Box H
Horsham, PA 19044
Phone: (215) 343-9018

SYMPOSIA SCHEDULING
James McGlinchey

(same address)

PRE-SYMPOSIA SEMINARS
Tom Viana

Naval Underwater Systems Center
Code 3521 Bldg. 1711-1
Newport, RI 02840
Phone: (401) 841-3354

PAPERS COORDINATOR
Bill Tippie

Argonne National Laboratory
9700 South Cass Avenue
Argonne, IL 60439
Phone: (312) 972-2000

PUBLICATIONS

PUBLICATIONS COORDINATOR
Robert Hassinger

Liberty Mutual Research Center
71 Frankland Road
Hopkinton, MA 01748
Phone: (617) 435-3452

MULTI-TASKER EDITOR
* Ralph Stamerjohn

Monsanto, Zone T1A
800 N. Lindbergh Blvd.
St. Louis, MO 63167
Phone: (314) 694-4252

HANDOUTS
Margaret Knox

(same as below)

QUESTIONAIRES
Unassigned

SOFTWARE

SOFTWARE COORDINATOR
* Phil Cannon

Science Applications Inc.
Suite 901
1211 West 22nd Street
Oak Brook, IL 60521
Phone: (312) 655-5960

TAPE COPY
* Jim Neeland

Hughes Research Laboratory
3011 Malibu Canyon Road
Malibu, CA 90265
Phone: (213) 456-6411 x333

DECUS LIBRARY
Glen Everhart

RCA, Government Systems Division
Route 38
Cherry Hill, NJ 08358
Phone: (609) 338-6022

PROJECTS

PROJECTS COORDINATOR
* Margaret Knox

University of Texas at Austin
Computation Center
Austin, TX 78712
Phone: (512) 471-3241

MENU COORIDNATOR
* Legare Coleman

Phillip Morris, U.S.A.
P. O. Box 26603
MS: JRC 43-2
Richmond, VA 23261
Phone: (804) 274-3251

SIG-FUNDED PROJECTS
Unassigned

USER GROUPS

USER GROUP COORDINATOR
Mike Fraser

Armed Forces Radiobiology Research
Biomathematics & Computer Support
Code BCS
Bethesda, MD 20814
Phone: (202) 295-1372

WORKING GROUP COORDINATOR
Tom Viana

Naval Underwater Systems Center
Code 3521, Bldg. 1711-1
Newport, RI 02840
Phone: (401) 841-3354

ALUG - EAST
Gregg Church

GEJAC, Inc.
P. O. Box 188
Riverdale, MD 20840
Phone: (301) 864-3700

ALUG - WEST
Bob Denny

Creative System Design
3452 East Foothill Blvd.
Suite 601
Pasadena, CA 91107
Phone: (213) 792-9474

WORKING GROUPS

VIRTUAL DISKS
DR. L. Michael Fraser

(same as above)

TRAINING
Tom Viana

(same as above)

DECUS LIBRARY
Glen Everhart

(same as above)

SIG TAPE COLLECTION
Jim Neeland

(same as above)

FILE SYSTEM AND I/O DRIVERS
Terry Medlin

NIH/NIDR
Bldg. 30 Room B23
Bethesda, MD 20014
Phone: (301) 496-1621

STRUCTURED FORTRAN
Bill Carroll

(in trasition)

SYSTEM PERFORMANCE AND ACCOUNTING
James Downward

KMS Fusion, Inc.
3941 Research Park Drive
Ann Arbor, MI 48104
Phone: (313) 769-8500

IAS
John Jenkinson

MOSTEK Corporation
P.O. Box 169
1215 West Crosby Road
Carrollton, TX 75006
Phone: (214) 323-6401

RSX-11M UNSUPPORTED VERSIONS
William Burton

Texas Research Institute
1300 Moursand
Houston, TX 77030
Phone: (713) 797-1976 x501

PROCESS CONTROL
Mark King

Colorado Interstate Gas
P.O. Box 1087
Colorado Springs, CO 80901
Phone: (303) 473-2300

SRD
Bob Turkleson

NASA/Goddard Space Flight Center
Mail Code 935
Greenbelt, MD 20771
Phone: (301) 344-5003

SPEAK OUT

"Speak Out" is a monthly column for readers to express their opinions or to comment on a previous column. The articles published in this column are an individual's viewpoint and do not necessarily reflect the opinion of DECUS or the RSX/IAS SIG. Readers are welcome to submit articles on any subject concerning the RSX/IAS world. Submissions may be editted by the Multi-Tasker staff for space considerations and clarity.

SOFTWARE MAINTENANCE SERVICES (Again)

Ralph W. Stamerjohn
Monsanto, St. Louis, Missouri

People were very outspoken at the Los Angeles DECUS Symposium on the subject of Software Maintenance Services. Sites had harsh words about Autopatch E and BRU problems, telephone support, and the policy of "will be fixed in the next release". I left the symposium feeling very uneasy and emotional about the whole issue.

But in retrospect, how bad is the problem? My systems have not crashed in over a year because of a RSX-11M system problem. BRU works better for me than any other magtape utility for software exchange and the one time we had a disk head crash, we did manage to recover the data from our BRU backup tapes after working some problems. Telephone support has answered some questions and SPR service has worked to fix other problems.

But I represent only one site. And my current systems for the most part are production systems that run the same code day after day. Are my experiences shared with the majority of RSX/IAS systems or are we the exception to the rule? Are the problems I do have shared with others or unique to my site?

Obviously, individuals have problems with Software Maintenance Services or there would not be complaints. The difficulty is measuring the impact and extent of problems when working with a small sample. To be able to work with Digital and solve problems with Software Maintenance Services, we need a broad base of information. At the end of this issue is a questionnaire to begin collecting information on Software Maintenance Services. It is critical all sites return this, otherwise, the only conclusions that can be drawn is that problems are limited to individual sites (or there is no interest in the user community in trying to solve the common problems). While these may very well be the correct answer, I would rather prove them by getting a deluge of responses.

We also want to study the effectiveness of the SPR mechanism. To do this, would all sites send copies of all raw SPR's they have submitted on RSX-11M V3.2, RSX-11M Plus V1.0, and IAS V3.0 and their layered products and any responses received from Digital. With this information, we can get a handle on how many (if any) unpublished patches exist, how well people write SPR's, and the volume we are dealing with.

Let us finally put the issue of Software Maintenance Services to rest. All software will have bugs in it so we need the services to work. And the better they work, the more time and money we will save.

FROM FIVE YEARS AGO

The December 1976 edition of the Multi-Tasker (Vol. 6, No. 5) was one of the longest, the following January issue (Vol. 7, No. 1) was one of the shortest. The December issue carried much feedback on the Atlanta symposium held in the Spring of 1976. The issue started with a letter from Carl Gibson, then Real-Time Product Manager, explaining Digital's current software warranty policies. Next, Carl responded to three SIG resolutions passed at Atlanta. The text of this response went as follows:

RESOLUTION

The RSX-11 SIG regards the change in Digital's policy to provide support for a previous release for 90 days after release of the succeeding release as totally inadequate and urges Digital to reconsider this policy.

Response

In recognition of the fact that Real-Time application systems, by their very nature, must be carefully integrated with the executive software, Digital will support superceded releases on the RSX-11S, RSX-11M, RSX-11D, and IAS Operating Systems for 6 months after customer availability of a new release. This exception to corporate policy is intended to answer the specialized needs of Real-Time users. In addition, every effort will be made to announce new release ordering information early enough to allow users to place orders in anticipation of release availability. This will shorten delays in obtaining the new release.

RESOLUTION

The RSX-11 SIG formally requests that a mechanism be set up so that a system manager can set any or all RSX-11D or RSX-11M MCR functions individually and in their various syntactical forms to be available to privileged, non-privileged, or not logged on terminals.

Response

RSX-11D and RSX-11M, as Real-Time Systems, have a number of shortcomings when used as Timesharing Systems. IAS was developed to meet these needs and includes control features of this type. Currently, Software Engineering efforts for RSX-11D and RSX-11M are focused upon a goal of making them better Real-Time Systems. We do not, at this time, plan to implement this feature in the upcoming system releases. The request will, however, be considered for implementation at a later time.

RESOLUTION

The RSX-11 SIG requests that Logical Device Assignment be implemented for RSX-11D in a functionally equivalent manner to the Logical Device Assignment facility of RSX-11M.

Response

We were not able to address this feature for Version 6.2 of RSX-11D. Our efforts for that release are directed entirely at System Reliability improvements. Work is being done in this area for IAS and we are considering adding the capability to RSX-11D if it can be provided without impacting system size or (more importantly) compatibility with the previous release.

However, the main part of the December edition was a list submitted by Eric Pollack of Digital of the DECUS/Atlanta wish list and the current resolution of each item as far as the author could determine. The list had 62 items and is must reading for any 'RSX' historian. By my quick count, 34 of the items will be implemented for IAS V3.1 and RSX-11M V4.0 and many of those not done had an explanation of why they were impossible.

Finally, the two issues published 10 SPR's for RSX-11D and five each for RSX-11M and IAS.

HELP YOURSELF

"Help Yourself" is a place for you to get your tough questions answered. Each month, questions from readers will be published. If you have a question, send a letter to the Multi-Tasker at one of the addresses listed on the cover.

We would also like to publish the answers to questions. If you can help someone, send a letter to the Multi-Tasker or call Ralph Stamerjohn (see first page). Your answer will be sent directly to the person in need and published in the next edition of the Multi-Tasker.

THIS MONTH'S QUESTIONS

RATFOR USERS

I would like to talk to users of RATFOR about your experiences, particularly people who are using for software delivered to customers and not just sites using it in-house.

C.W. Holeman, Staefa Control System, 4340 Viewridge, San Diego, California. Phone (714) 571-7771.

DECNET PHASE II PROBLEMS

I would like to talk to anyone who is still using DECNET Phase II under RSX-11M V3.2. We have some intermittent, unreproduceable problems and would like to compare notes with you.

Ken Johnson, Monsanto, Zone T1D, 800 N. Lindbergh, St. Louis, Missouri, 63166. Phone (314) 694-4480.

GOULD ES1000 SUPPORT

Does anyone have or know of subroutines to support the Gould ES1000 electostatic recorder?

Randy Biallas, Medtronic, Minneapolis, Minnesota, 55440. Phone (612) 574-3623.

HP 9874/7221 SUPPORT

Does anyone have or know of subroutines to support the HP 9874 digitizer or the HP 7221 plotter?

Randy Biallas, Medtronic, Minneapolis, Minnesota, 55440. Phone (612) 574-3623.

VT-103 PRINTER/PLOTTER SUPPORT

Can anyone provide information on a small (6"x4"x3") printer/plotter that needs to fit inside a VT-103? We need about 100 dots/inch resolution, i.e. ECG strip chart with annotation.

Randy Biallas, Medtronic, Minneapolis, Minnesota, 55440. Phone (612) 574-3623.

RSX-11M/RSX-11M PLUS DRIVERS FOR DT07

I would be pleased to hear from anyone that has a driver for the DT07 Bus Switch for either RSX-11M or RSX-11M Plus. The application is for a single CPU with the bus being manually switched to a backup, non-coupled CPU.

Phillip W. Rowland, Celanese Corporation, User Technical Services - 183, P.O. Box 32414, Charlotte, North Carolina, 28232. Phone (704) 554-2100.

ENGLISH/JAPANESE TERMINAL

I am looking for an interactive terminal that will handle both the English language and all three (3) of the Japanese character sets (Katakana, Hiragana, and Kanji). Anyone knowing of such a device, please contact me.

Russell Dee Rolfe, Japanese/English Project Manager, Weidner Communications, 1673 W. 820 North, Provo, Utah, 84601. Phone (801) 375-9910. TWX 453074.

TU58 PROBLEMS

A recently purchased 11/34 system included a TU58 DECtape II for cheap offline and archival storage. The device was also intended as a software transfer medium from an existing 11/34 system using an extra DL11-W on the old system. The new system included the standard DL11-E interface for the TU58. The unit worked great on the new system as either a Files-11 or RT device (PIP or FLX) but numerous problems were encountered when using the TU58 on the old system using either the DL11-W or the new DL11-E. Part of the problem seemed to be the loss of characters from the TU58 at 9600 baud. Reducing the rate to 4800 greatly reduced but did not eliminate the problem and 2400 baud was no better than 4800. Since the TU58 does not belong to the old system, I cannot get hardware support. I am interested in hearing from anyone who is using a TU58 in a configuration other than a PDP-11/44, especially anyone using it with a DL11-W.

Leo C. Young Jr., Bell Helicopter Textron, P.O. Box 482, Department 87, Fort Worth, Texas, 76101. Phone (817) 280-5168.

RSX-11M TERMINAL I/O PERFORMANCE

After examing the performance of our RSX-11M installation (PDP-11/44, 128 KW), we would like information on the effect and implementation on the following modification which we feel are necessary to improve performace:

1. Programs doing terminal output should not automatically checkpoint but should allow other tasks to cause checkpointing according to normal priorities. Checkpointing should be forced if the program has an output time significantly longer than the swap time or is held up waiting for a control-Q.

   This is to get adequate output rates from high priority programs when the system is swapping, without allowing tasks doing output to delay higher priority tasks.

2. Programs doing terminal input should not be forced to checkpoint immediately but should wait some user-settable period to see of the request is satisfied before being forced to checkpoint. During this wait period, other tasks could cause checkpointing according to the normal priorities.

   This is to ensure that terminal input can proceed at normal rates and an adequate response time for tasks with sufficient priority. This feature is particularly important for KED and word processors.

3. All input should be buffered so that type-ahead is always available. Due to slow response times, input may go to MCR instead of the program that has just prompted for a response.

4. The type-ahead buffer should be increased in size to at least 80 characters. When type-ahead feature is activiated by a program, i.e. KED, the current buffer length is easily exceeded.

5. Limited tests with 256 KW of memory indicates swapping overhead increased by a factor of two to four times. Can swapping overhead be reduced as memory increases?

W.J. Whiten, Julius Kruttschnitt Mineral Research Centre, Isles Road, Indooroopilly, Brisbane 4068, Australia.

READING BRU TAPES ON DEC-10'S

I am running RSX-11M V3.2 on a RL02 system with a single TE10 tape drive (800 BPI only), and have difficulty copying SIG tapes because of the small disks. We also have a DECsystem-10 with lots of tape drives and disks and I have been successful in copying the FLX versions of the SIG tapes on that machine. Does anyone know of an utility that will copy BRU tapes on the DECsystem-10?

Ronald G. Browne, PFIZER Inc., Pfizer Central Research, Groton, Connecticut, 06340. Phone (203) 445-5611, x673.

HINTS AND THINGS

"Hints and Things" is a monthly potpouri of helpful tidbits and rumors. Readers are encouraged to submit items to this column. Any input about any way to make life easier on RSX/IAS is needed. Please beware that items in this column have not been checked for accuracy.

RMS-11K WORK-AROUNDS

Robert F. Thomas
A.S. Thomas
Westwood, Massachuesetts

We have been using Fortran-IV Plus V3.0 since its initial availability. We have extensively used the RMS-11K support in this version and over this period, have discovered several problems with RMS-11K and various work-arounds:

1. The TKB option MAXBUF must be set equal to the largest bucket size of any file to be opened by the program. Example:

   Assume the program uses file XYZ.RMS which has an RMS bucket size of two (2). This means there are two blocks to a bucket and MAXBUF=1024 (2*512).

2. Mysterious TKB errors and strange operating behavior occur if all files are opened in overlays. This problem is generally exhibited by tasks which, when built, issue the following warning one or more times:

   TKB -- *DIAG* - n UNDEFINED SYMBOLS SEGMENT xxxx $RMREC

One method of getting around this problem is to include a dummy reference to $RMREC in the root segment. The method we frequently use is to include a dummy OPEN statement in the root segment which uses the maximum functionality required by any routine performing an OPEN in the program.

3. When reading an RMS-11K indexed file using a key, the key length must be specified at compile-time. This contradicts the documentation on page 7-6, paragraph 7.2.4.1 of the PDP-11 Fortran Language manual. One way to perform generic reads is to write a general function where the ket value is passed as a dummy array whose size is passed as a dummy variable.

RSX PRODUCT PANEL

Ralph Stamerjohn
Multi-Tasker Editor

From the Editor

This is a short report on the RSX Product Panel given by Steve Paavola at the Los Angeles Symposium on Thursday night, before the RSX-11M Question and Answer Session. It is taken from my rough notes and should only be used for informational purposes.

At the RSX Product Panel, RSX-11M V4.0 and RSX-11S V4.0 scheduled release dates were announced as March/April. RSX-11M Plus V2.0 is currently scheduled for release in May of 1982. The media requirements for RSX-11M V4.0 for the various kits is expected to be as follows:

RK05	11 disks	(9 RSX-11M, 1 RMS, 1 Autopatch)
RL01/2	7 disks	(5 RSX-11M, 1 RMS, 1 Autopatch)
RK06/7	4 disk	(2 RSX-11M, 1 RMS, 1 Autopatch)
RPxx	800 BPI Tape Kits	(4 tapes and DSC tape)
RPXX	1600 BPI Tape Kits	(3 tapes and DSC tape)

RSX-11M Plus will have two distribution forms: a one RL02 disk with a runnable system requiring no generation or three tapes and a DSC tape.

RMS-11, including index ISAM support, is bundled into the next release. However, the RMS portion of the kits will be distributed separately when RMS V2.0 is release.

For the next release after RSX-11M V4.0, Digital is planning one major change and took a poll to measure its impact: system generation will require a mapped

machine with 64 KW's at least. Smaller target systems, including unmapped systems, could still be generated. This is not a firm announcement, so if this will cause you problems, contact Steve Paavola (address in previous article). No attendees at the session voiced any concern.

Similarly, a poll was taken to drop support for old devices two releases from RSX-11M V4.0. The list of devices included the RF11, RP02, RP03, TA60, DECtape, VT11, DP11, UDC-11, AFC-11, AD01, and LPS-11. It is expected that the old device drivers will continue to work, however, no support is planned. Again, contact Steve Paavola if these plans present a problem. The mininum time for this action is two years.

IAS QUESTION AND ANSWER SESSION

Terry Bossert
Paul D. Clayton

Republic Management Systems
Trevose, Pennsylvania

Steve Citko, Milan Merit ICG RR

Q: Having some problems with FORTRAN 77. Heard problems with task builder working with FORTRAN 77. Do we need a new task builder?
A: Changes should be in IAS 3.1.

Q: Have experienced FORTRAN virtual array degradations of up to 20 or 30 times. Is that to be expected?
A: No.

Comment on AUTOPATCH: People are getting patches from TSC which do not get published or included in AUTOPATCH. Therefore many different versions of IAS with minor variations are emerging. Wouldn't it be more logical to have one master copy of IAS sitting somewhere containing all the patches (which is rereleased twice a year instead of AUTOPATCH or more often if requested).

Paul D. Clayton, Republic Management Systems.

Q: Are the normal device handlers capable of running IOX, or do I need the handlers from the IOX account on the IAS distribution tapes?
A: All the normal device drivers are good for IOX. Do not use the IOX handlers.

Q: I have a user written null task run at priority 1 before timesharing is brought up. After some time the task gets on the MRL and never gets off. Any ideas as to why.
A: No suggestion to look at.

Q: Is there any way to have multiple copies of a device handler service same device types on multiple device interfaces?
A: Audience - There is a way to do it. The procedure will appear in the DeVIAS newsletter.

Q: Can a list be published of those tasks that assume device characteristics by the device name?
A: Yes.

Robert F. Curley, University of Pennsylvania

Q: Is there a place that we can find the required hardware revision levels necessary to run IAS 3.1, so that we could help inform our field service people?
A: We would go to field service ourselves, but we will ask them and try to publish it in the Software Dispatch.

Q: It has been rumored that DECNET Phase III requires 280 nodes while resting. Would you please comment?
A: DECNET should use the same as last time. It depends upon configuration of the DECNET system.

Q: Could you give us a hook in LOGIN to run our own task?
A: Yes, we could publish a patch in the Software Dispatch.

Q: Is it possible to build a bootable task via VMR under IAS? A user written task that can be built and loaded on mag tape that can be booted?
A: We build DSC under RSX-11S as a stand alone task and use VMR to copy it to tape. You could use that too if you have RSX-11S. But you can not do it under IAS.

Larry W. Ebinger, Sandia National Labs

Q: FMT does not work with DKOVL.TSK for RK05 disk drive. Disk drives were upgraded from RK05's to RL02's. FMT does not work with RL02 disk drives. What gives?
A: We are looking at it.

Q: Why does DEC patch a module instead of giving a new module?
A: So that modules patched from the Software Dispatch will have the same checksum as those from the AUTOPATCH.

Q: Will a system saved with DECNET running, boot if some devices are physically removed from the system (ie. removing a DMC)?
A: Don't know at this time. Check with me later on this week.

Alan Frisbie, Flying Disk Systems

Q: Exactly what is the incompatibility between the patched and unpatched versions of BRU (IAS 3.1 release notes)?
A: Not really known, but might have something to do with the maximum number of files permitted on a volume.

Michel Gauthier, Service Tele Informatique Au Saguenay

Q: Has anybody tried to generate IAS on an 11/24?
A: No answer.

Ronald Griffith, University of Michigan

Q: When running several heavily overlayed programs which fill core, I at times, have system crashes. They occur most frequently when I'm using the TU16 or running 2780 RJE. The CPU actually halts, according to the front panel. Have others had this problem?
A: No. Suggest you restart the system at address 44 and attempt a core dump for analysis.

Ken Guralnik, E.G. & G.

Q: Whenever our dual RP06 drives are simultaneously used (i.e. DSC from one to another) on our 11/44, the system hangs up. It appears interrupt enable is reset. Is this software or hardware problem?
A: Probably hardware.

Tom Mathieu, Battelle NW

Q: Using BRU, can you copy files from a mounted disk.
A: Yes, mount them with DCF.

Q: Mounting a tape on MT with wrong label or off-line generates a "SYSTEM ERROR 3" or "SYSTEM ERROR 4"
A: No answer.

Janet McCormick, American College of Radioloy

Q: With version 3.1 of IAS we have not been able to user our RX01 floppies successfully. The problem is intermittent. If we can mount the DX, we hang the system when we try to access it. Sometimes the system hangs when we mount it.

A: The problem is the handler for the DX. If you rebuild the DX handler from IAS 3.0 the problem will go away.

George Rhodes, Bunker Ramo Corporation

Q: We forced crash of the system, pressed the 'CONT' on the front panel and got a dump of the system to the magtape unit. The tape ran forware and then rewound for up to 1.5 hours before I halted the CPU. Why?
A: It is believed that a bad spot on the maptape will cause this to happen. Errors during a system dump are not recoverable.

Q: Pre-release of 3.1 DSC had a blocking factor that worked fine; the official release has not provided as good throughput regardless of the blocking factor.
A: DEC- the pre-release version of DSC was dangerous due to a user not specifying the same blocking factor when going to tape from disk and back again. Audience - no problem found with the official DSC release and is faster and does recognize blocking factor.

S. Strezleck, Nabisco Brands

Q: Asked for examples of how to build tasks with memory resident overlays.
A: Suggestion was to look at IAS system tasks as examples.

Mark Wiederspahn, University of Texas

Q: Is RESFCP supported?
A: Yes.

Q: Do you have any performance guidelines or benchmarks between BIGFCP and RESFCP?
A: No.

Q: SHOW MEMORY shows a task in memory. SHOW TASK,,ALL shows no task active. Why the discrepancy?
A: SHOW MEMORY scans the memory usage list, while SHOW TASK,,ALL scans the ATL. Perhaps task is in the 'MRR' state.

Q: How can I redirect batch logs so they are not printed.
A: Audience - rename [1,4]BATCH.LOG to anything else. The despooler will not find it then, since it knew the original name. The renaming must be done within the batch stream itself.

Q: Sometimes a task running in level 3 (compute bound) at the same time as a batch task is running gets 'stuck' and the system won't run anything but the level 3 task. Batch quantum = 2.
A: Batch quantum is too small, increase it.

Mike Yankus, Planning Research Corp.

Q: If the SY: disk momemtarily down, attempting to use MO.... leads to system crash. TKTN was not fixed. CDA showed that MO.... apparently aborted. Because TKTN could not be loaded, no print out was directed to TT0:.
A: 1) MO.... is overlaid. Load failure of overlay causes MO.... to abort. Handlers aborting is always trouble. 2) All termination notices are issued by TKTN, none are done by the IAS executive. The exec does not notify anyone of load failures either.

Q: Why doesn't SYSBLD.CMD fix (or recommend fixing) TKTN as was done with RSX-11D?
A: No reason why TKTN cannot be fixed. No one remembers why the fix for TKTN was removed from SYSBLD.CMD.

FORTRAN INTERFACE TO UNIVERSAL LIBRARY FILES

R. N. Stillwell

Baylor College of Medicine
Houston, Texas

A group of files which are frequently accessed together, but infrequently updated, is conveniently maintained as a library file. IAS and RSX use libraries principally for object modules and for macros; however under current versions of these systems users can create "universal libraries" to store groups of files of similar type. Documentation on universal libraries is found in the chapters on LBR of the IAS and RSX-11 Utilities Manuals, and in Appendix B of the IAS/RSX-11 System Libraries Reference Manual (Update 1).

There are a couple of problems associated with the use of the $ULA routine described in the System Libraries manual which are not apparent from the documentation; these have to do with the proper detection of the end-of-file in the currently open module. One of these problems was solved empirically by doing a .MARK and .POINT as indicated below; perhaps a reader familiar with FCS can suggest a better solution.

The following MACRO code provides a Fortran-callable interface to a universal library; it also provides Fortran access to the .MARK and .POINT routines, allowing the Fortran programmer to switch a LUN to a different input file, then return to the original input file at the same point. This code was developed and tested under IAS version 3.1 and Fortran IV version 2.5. Presumably it should work with RSX-11M or Fortran IV-Plus.

```
        .TITLE  ULBIO
;
;   SUPPORT FOR FORTRAN ACCESS TO A UNIVERSAL LIBRARY;
;   SUPPORT FOR .MARK AND .POINT FOR FORTRAN.
;
; FORTRAN-CALLABLE ROUTINE TO ISSUE A $ULA CALL.
;
; THE UNIVERSAL LIBRARY FILE MUST BE OPENED FIRST, READ-ONLY.  THIS CAN BE
; BY MEANS OF CODE SIMILAR TO THE FOLLOWING:
;
;       INTEGER ULBOPN,ULBCLS
;       INTEGER BUFFER(7), MODNAM(2)
;       DATA LUN /1/
;C  LUN IS AN INTEGER, LOGICAL UNIT NUMBER
;C  MODNAM IS A TWO-WORD RAD50 MODULE NAME
;C  BUFFER IS A 7-WORD BUFFER FOR STORING THE FIRST 7 WORDS OF THE FDB
;C  IRET IS RETURN CODE: SHOULD BE +
;C
;C  OPEN LIBRARY FILE, READ-ONLY
;C
;       OPEN (UNIT=LUN,NAME='LIBRARY.ULB',ACCESS='SEQUENTIAL',
;      1 TYPE='OLD',READONLY[,FORM='UNFORMATTED'])
;C
;C  CONVERT MODULE NAME TO RAD50 AND OPEN THE MODULE
;C
;       CALL IRAD50(6,'MODUL1',MODNAM)
;       IRET = ULBOPN(LUN,MODNAM,BUFFER)
;       IF (IRET.LT.0) STOP
;
;C  SUBSEQUENT READS TO THE LUN READ FROM THE MODULE JUST OPENED.  FORMATTED
;C  OR UNFORMATTED READS MAY BE USED, AS APPROPRIATE.
;C
;C  BEFORE CALLING ULBOPN AGAIN TO OPEN A DIFFERENT MODULE, OR BEFORE CLOSING
;C  THE FILE, ULBCLS MUST BE CALLED.  BUFFER MUST NOT BE ALTERED BETWEEN
;C  CALLS TO ULBOPN AND ULBCLS.
;
;       IRET = ULBCLS(LUN,BUFFER)
;
;
        .PSECT  ULBIO,RO,I,CON,LCL,REL
```

```
ULBOPN::
        MOV     @2(R5),R2               ;GET LUN
        MOV     @#$OTSV,R3              ;ADDRESS WORK AREA
        CALL    $FCHNL                  ;FFDB ADDRESS IN R0
        BCS     ERROR
        ADD     #14,R0                  ;ADDRESS FCS FDB
        MOV     4(R5),R1                ;POINT TO RAD50 NAME
        MOV     (R1)+,NAMBUF            ;MOVE TO BUFFER
        MOV     @R1,NAMBUF+2
        MOV     #NAMBUF,R1              ;POINT TO BUFFER
        MOV     6(R5),R2                ;POINT TO BUFFER FOR ORIGINAL FDB
        MOV     #7,R3                   ;COUNT 7 WORDS
        MOV     R0,R4                   ;COPY FDB ADDRESS
1$:     MOV     (R4)+,(R2)+             ;MOVE
        SOB     R3,1$
        MOV     F.URBD+2(R0),-(SP)      ;SAVE BUFFER DESCRIPTOR
        MOV     F.URBD(R0),-(SP)
        MOV     #100,F.URBD(R0)         ;PUT SIZE OF NAMBUF-4 IN FDB
        CALL    $ULA                    ;OPEN THE MODULE
        BCS     FILER2
        MOV     F.VBN+2(R0),BEGBLK      ;SAVE STARTING VBN (L.O. ONLY)
;
                                        ;THIS PECULIAR CODE SEEMS TO BE
        CALL    .MARK                   ;  NECESSARY IF A SUBSEQUENT READ
        BCS     FILER2                  ;  IS TO RECOGNIZE AN EOF OCCURRING
        CALL    .POINT                  ;  IN THE FIRST BLOCK OF THE MODULE
        BCS     FILER2                  ;  AND NOT GO ON READING TO THE END
                                        ;  OF THE BLOCK.
;
                                        ;IF THE FOLLOWING CODE IS NOT INCLUDED
                                        ;  A .POINT TO A BLOCK > 1 OF THE
                                        ;  MODULE WILL FAIL.
        MOV     F.EFBK(R0),F.HIBK(R0)   ;SET HIGHEST VB = EOF BLOCK
        MOV     F.EFBK+2(R0),F.HIBK+2(R0)
;
        MOV     (SP)+,F.URBD(R0)        ;RESTORE BUFFER DESC.
        MOV     (SP)+,F.URBD+2(R0)
        BR      FILERR                  ;GET RETURN CODE AND RETURN
;
ERROR:  MOV     #-1,R0                  ;ERROR RETURN
        RETURN
;
FILER2: ADD     #4,SP                   ;CLEAN UP STACK
FILERR: MOVB    F.ERR+1(R0),R1
        MOVB    F.ERR(R0),R0
        RETURN
;
ULBCLS::
        MOV     @2(R5),R2               ;GET LUN
        MOV     @#$OTSV,R3              ;ADDRESS WORK AREA
        CALL    $FCHNL                  ;FFDB ADDRESS IN R0
        BCS     ERROR
        ADD     #14,R0                  ;ADDRESS FCS FDB
        MOV     4(R5),R1                ;POINT TO BUFFER TO RESTORE
        MOV     #7,R2                   ;COUNTER
```



```
2$:     MOV     (R1)+,(R0)+             ;MOVE 7 WORDS
        SOB     R2,2$
        CLR     R0
        CLR     R1
        RETURN
;
        .PSECT  ULBDAT,RW,D,OVR,GBL,REL
NAMBUF: .BLKW   42.                     ;BUFFER FOR NAME AND MODULE HEADER
BEGBLK: .WORD   0                       ;FOR STARTING BLOCK
;
        .PSECT  ULBIO,RO,I,CON,LCL,REL
;
;  FORTRAN INTERFACE TO MARK AND POINT ROUTINES, AND TO REWIND AND FIND
;  END OF FILE.
;
;  FMARK, FPOINT, AND FEOF WORK WITH ANY SEQUENTIAL FILE.  ULBRWD REWINDS
;  A MODULE WHICH HAS BEEN OPENED WITH ULBOPN.
;
;       INTEGER FPOINT, FMARK, ULBRWD, FEOF
;
;       IRET = FMARK(LUN,POSIT)
;
;C  LUN IS UNIT NUMBER
;C  POSIT IS 2-WORD BUFFER TO RECEIVE CURRENT BLOCK AND BYTE OFFSET
;
;       IRET = FPOINT(LUN,POSIT)
;
;C  LUN IS UNIT NUMBER
;C  POSIT IS 2-WORD BUFFER CONTAINING BLOCK AND BYTE OFFSET
;
;       IRET = ULBRWD(LUN)
;
;C  REWINDS CURRENT LIBRARY MODULE
;
;       IRET = FEOF(LUN)     POSITION FILE TO EOF
;
;
FMARK::
        MOV     @2(R5),R2               ;GET LUN
        MOV     @#$OTSV,R3              ;ADDRESS WORK AREA
        CALL    $FCHNL                  ;FFDB ADDRESS IN R0
        BCS     ERROR
        ADD     #14,R0                  ;ADDRESS FCS FDB
        CALL    .MARK
        BCS     FILERR
        MOV     4(R5),R1                ;ADDRESS OUTPUT BUFFER
        MOV     R2,(R1)+                ;RETURN L.O. BLOCK NUMBER
        MOV     R3,(R1)+                ;RETURN BYTE OFFSET
        CLR     R0
        CLR     R1
        RTS     PC
;
FPOINT::
        MOV     @2(R5),R2               ;GET LUN
        MOV     @#$OTSV,R3              ;ADDRESS WORK AREA
```

```
        CALL    $FCHNL                  ;FFDB ADDRESS IN R0
        BCS     ERROR
        ADD     #14,R0                  ;ADDRESS FCS FDB
        MOV     4(R5),R1                ;ADDRESS INPUT BUFFER
        MOV     (R1)+,R2                ;L.O. BLOCK NUMBER
        MOV     @R1,R3                  ;BYTE OFFSET
        CLR     R1                      ;H.O. BLOCK NUMBER
        CALL    .POINT
        BR      FILERR                  ;GET RETURN CODE
;
ULBRWD::
        MOV     @2(R5),R2               ;GET LUN
        MOV     @#$OTSV,R3              ;ADDRESS WORK AREA
        CALL    $FCHNL                  ;FFDB ADDRESS IN R0
        BCS     ERROR
        ADD     #14,R0                  ;ADDRESS FCS FDB
        MOV     BEGBLK,R2               ;L.O. BLOCK NUMBER
        CLR     R3                      ;BYTE OFFSET
        CLR     R1                      ;H.O. BLCOK NUMBER
        CALL    .POINT
        BR      FILERR                  ;GET RETURN CODE
;
F      ::
        MOV     @2(R5),R2               ;GET LUN
        MOV     @#$OTSV,R3              ;ADDRESS WORK AREA
        CALL    $FCHNL                  ;FFDB ADDRESS IN R0
        BCS     ERRORA
        ADD     #14,R0                  ;ADDRESS FCS FDB
        MOV     F.EFBK(R0),R1           ;HIGH ORDER BLOCK NUMBER
        MOV     F.EFBK+2(R0),R2         ;LOW
        MOV     F.FFBY(R0),R3           ;NEXT BYTE
        CALL    .POINT                  ;POINT TO EOF
        JMP     FILERR
;
ERRORA: JMP     ERROR                   ;BRANCH AID
;
        .END
```

MULTIPLE WRITERS TO FCS FILES

Ken L. Johnson

Monsanto
St. Louis, Missouri

This paper describes block locking in RSX-11M V3.2 and explains how to use block locking with the FCS file access routines to allow multiple writers to a file.

1.0 WHAT IS BLOCK LOCKING?

Block locking support exists in the RSX-11M executive and in F11ACP for support of RMS features. However, this feature can also be used with FCS.

It is possible to open a file with multiple writers with FCS. The difficulty is synchronizing access to blocks in the file. How can read-modify-write cycles in separate tasks be prevented from overlapping? Block locking can provide a guaranteed exclusive read-modify-write cycle.

Block locking can be activated when a file is opened, and remains on until the file is closed. If block locking has been turned on, then blocks are locked whenever they are accessed with a read or a write.

Block locking works on a per lun basis. That is, a block locked by lun 1 in a task cannot be accessed on lun 2 of the same task. The error returned is IE.LCK. Once locked, access on all other luns in all tasks is prevented until the block is unlocked.

Blocks can be unlocked automatically by the executive or with a QIO issued by the program. Blocks which were locked by a QIO which is still in progress cannot be unlocked. This applies to unlock QIOs (IO.ULK) and to the automatic unlocking done by the executive. Thus an unlock QIO issued before the previous read QIO has completed could not unlock the blocks locked by the read QIO. The error code returned is IE.ULK. All blocks locked by a task are always unlocked when the task closes the file or exits.

The choice of automatic or manual unlocking is made when the file is opened. If automatic unlocking is selected then all currently locked blocks are unlocked when a new block is accessed, provided that I/O is complete. When manual unlocking is selected, blocks are unlocked only when the task issues the special unlock QIO, or when the file is closed.

2.0 AN APPLICATION EXAMPLE

Block locking is used extensively in one of our applications at Monsanto. The data for this application is stored in one large file. This file has an internal structure known to the application. Blocks within the file are allocated from and deallocated to an internal free list. The integrity of the structure and of the free list listhead are guaranteed through the use of block locking. Application programs in the system use common subroutines to open and close the file, and to do reads and writes. A subroutine is also available to unlock blocks in the file. The file is opened in one of three ways: automatic unlock, manual unlock, and read-only. When the file is opened read-only, it is possible to read blocks without locking them. This is often useful.

3.0 HOW TO USE IT

The first requirement to use block locking is that the support be generated into your system. To include block locking support, answer yes to the Sysgen Phase I question "RMS record locking and placement control?", which is question 3 in the Executive Options. This option enables the executive support. F11ACP is supplied by DIGITAL with the necessary support, it does not need to be reassembled.

It is very easy to enable block locking. All that is required is to set up the upper byte of F.ACTL prior to the open. For automatic unlock, use:

```
        MOV     #FA.ENB!FA.DLK!FA.EXC,F.ACTL(R0)
```

For manual unlock, use:

```
        MOV     #FA.ENB!FA.DLK!FA.EXC!FA.RWD,F.ACTL(R0)
```

where R0 points to the FDB in both cases. These bits are all defined in the macro FCSBT$. The definitions are

```
        FA.ENB = 100000
        FA.RWD =   4000
        FA.EXC =   2000
        FA.DLK =   1000
```

FA.ENB enables the the access control word F.ACTL. FA.EXC turns on executive block locking. FA.RWD (mag tape rewind bit) specifies manual unlocking. FA.DLK turns off the lock bit on the file, which is necessary to allow multiple writers.

The unlock QIO is also straight-forward. The I/O function code is IO.ULK. The user must specify the lun that the file is open on. The QIO parameters are:

1. Unused
2. (Number of blocks to unlock)*512. or zero
3. Unused
4. High byte unused, low byte is high part of VBN
5. Low part of VBN
6. Unused

In the discussion of unlocking which folllows, remember that a block locked by a QIO in progress cannot be unlocked by another QIO.

An exact match must be found if the VBN and byte count are specified. The byte count is rounded up to the next block boundary. If an exact match does not exist, IE.ULK is returned.

All currently locked blocks on this lun are unlocked if the VBN and byte count

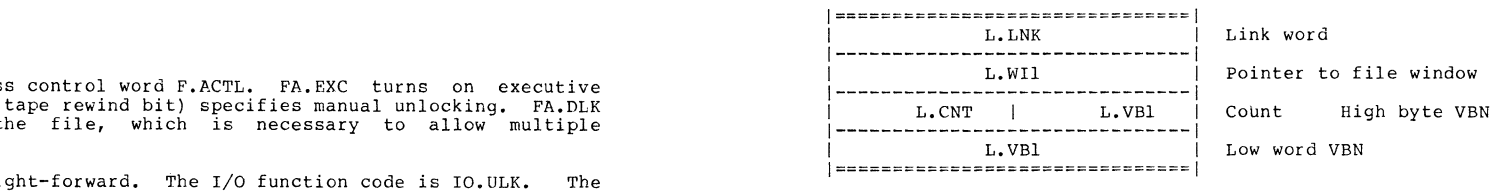
parameters are zero.

The first lock list entry which matches the VBN will be unlocked if no byte count is specified. If two lock requests with different lengths and the same starting block number have been issued, one of them will be unlocked. The order of lock list entries is unpredictable because the entries can be reused.

4.0 THEORY

Block locking is implemented by the RSX-11M executive and F11ACP. Locking is implemented using a lock list in executive pool. This lock list is pointed to by offset W.LKL in the file window, and by F.LKL in the FCB. The lock list entry offsets are defined in the macro F11DF$. The code for block locking is conditionalized on the symbol R$$LKL. The two modules in the executive which contain this code are DRQIO and IOSUB. Read the subroutine $LCKPR in the module IOSUB for the details of lock processing.

The lock list entries are linked in a single list for each file. The file window and the FCB point to the beginning of the list.



The lock list entries are allocated from system pool by the executive as needed. The are deallocated by F11ACP when the file is closed. The executive unlocks blocks by clearing L.CNT. An entry with L.CNT = 0 can be reused. Whenever a IO.RVB or IO.WVB is issued to a file the executive checks the lock list, if any, for that file. This checking is done in the $LCKPR routine in IOSUB.

5.0 HINTS AND TRICKS

A QIO IO.RVB or IO.WVB to VBN 0 will fail with a bad parameters error. However, if block locking is turned on block 0 will be left locked. If the length to read or write was longer than one block, then some blocks will be left locked even though the read or write failed.

Watch out for programs which try to lock all of the blocks in a large file. That is a good way to run your system out of pool.

To unlock a block without aborting a task, use the OPEN command and zero the count in the appropriate lock entry.

REESE'S PIECES

Frank R. Borger

Micheal Reese Medical Center
Chicago, Illinois

Reese's Pieces is a collection of programs originally submitted to various SIG tapes by the Department of Medical Physics of Micheal Reese Medical Center. Although some of the programs are IAS specific (the current distribution is updated to IAS version 3.1), many of the programs are equally at home on an RSX-11M system. Some of the programs are not REESE originals, but are SIG tape programs modified by us.

The preparation of a separate distribution was made necessary by our own inability to keep track of where our various submissions were on the numerous SIG tapes and the obviously occuring conflicts when we tried to submit new versions. The programs are available by sending either 2 RK05 disks or a 9-track 800 BPI magtape to us. Please include a return mailer.

Contact: Frank R. Borger
Micheal Reese Medical Center
Department of Medical Physics
29th Street and Ellis Avenue
Chicago, IL, 60616

Phone: (312) 791-2515

A capsule description of the collection follows:

1. RSX BASIC
   This is a version of DOS BASIC, updated to include extensive string manipulation, full file access (random, sequential, fixed length, shared), and virtual arrays. It does some pre-compilation for faster operation, although it still is an interpreter. The interpreter requires FPP isntructions. Machine language subroutines are permitted. Versions for IAS and RSX-11M are supplied.

2. INFORM
   INFORM tells you everything about an IAS system you wanted to know, but SYS won't tell you. INFORM displays task status in various forms, terminal status, device forms types, node pool usage, and many other features.

3. DISOBJ
   DISOBJ takes a stock object module and reverse assembles it back into pseudo Macro-11 listing format. Global definitions are decoded and inserted into the text and the ASCII and RAD50 conversions of the code are printed in the "comment" field. This program is of high interest to all machine language gurus.

4. RUNOFF based programs

   * RNP
     This is a RUNOFF preprocessor which lets you merge variable input from the terminal and fixed input from secondary files into one master output file for later processing by RUNOFF. This is extremely good for producing letters and memos.

   * REESE RUNOFF
     This is a version of RUNOFF with some minor modifications for Centronics output and major modifications to support DIABLO-type printers, including proportional character spacing.

   * DIABLO
     DIABLO is a program to output REESE RUNOFF output directly to a letter-quality printer in write-pass-all mode so the control characters are not dropped by the terminal handler.

5. ABSZAP and CORZAP
   These are one-line versions of OPE which accept new core contents from the MCR command line so that they can be evoked by indirect command files for system patching, especially designed for system startup patching.

6. DAMMIT
   A frustration release mechanism which responds with random smart remarks.

7. DSKFIX
   A version of Glenn Everhart's program for absolute disk block patching. This is updated to handle large disks and capable of calculating block checksums for file-header blocks.

8. FAC
   A program to show the current open files on a disk.

9. FDU
   A program to dump a file simultaneously in three modes: ASCII, octal, and RAD50.

10. FHD
    A program to list the file-header block for a given file. It shows the full contents, including the block allocation and file-header extension blocks.

11. FRAG
    A program to display disk fragmentation. FRAG lists the number of free blocks, ordered by the size of the holes.

12. MANUAL
    An introductory manual for an IAS system. Designed to compress the basic information scattered around in 9 or 10 ring binders to a basic minimum needed to get a new user on the air.

13. MASSAGE
A catch-all task that receives command lines from MCR when it does not find the requested task installed. MASSAGE changes the command line into an appropriate form and submits it to PIP, QUE, or whomever should get it.

14. LUT
A utility to list the logical unit table for an active task, showing files that are opened.

15. PAG
A program to produce pageinated listings of data files, basic sources. It includes a banner page specific to Micheal Reese, but this can easily be modified for other sites.

16. RECOVER
The Boeing file recovery program, updated to handle multi-header files.

17. ROLLIN
An on-line version of DIGITAL's old ROLLIN program. Provides RK05 to/from tape backup and restore and disk-to-disk copy of disks in general.

18. DIALSERVE
Scans dial-up lines and logs user off and hangs line up if carrier is lost. PDX does this, but not for a MCR-based system.

19. WHO
Shows who is logged on at each terminal, their UIC and device, and any active tasks.

20. DEVPAT and SDV
These allow one to change their default device when running in MCR mode.

21. MCR, HEL, and BYE
These are modifications to allow PDS-style login and some user protection when running in MCR mode.

22. PRO and UNP
These are programs to protect (disable all delete access) and unprotect a file. These functions are now included in MASSAGE.

23. UTX
A program to display on a CRT tasks at each level of the time-sharing scheduler. You can see how tasks are being promoted and demoted. Also UTX indicates which tasks are being swapped and reloaded.

GENERAL PURPOSE ERROR HANDLING

D.B. Curtis

Science Applications, Inc.
Oak Brook, Illinois

From the Editor

This submissions is a piece of RSX MAGIC, but due to its length, is published as an article. It uses the AST mechanism Dan described in the October 1981 issue in the "From the Wizards Book of Magic" column. Dan also submitted his entire source code. This is not published due to its length, but can be obtained by writing the Multi-Tasker. A paper copy is available or I can supply machine-readable versions on either magtape or RX01's if you send a blank media and a return mailer.

This task provides a very efficient, general purpose, error display task. There is only a six word memory overhead at each error occurrence. Essentially this is done by dynamically adding an error directive to the executive. The technique indicates how to expand the functionality of the RSX-11M executive without modification to the executive source.

This particular version of the error task was written for an RSX-11S system. Therefore, I did not use disk resident error messages. However, it would be simple to add that capability. An example of the use of the error processor is shown below:

```
        MRKT$S  #1,#1,#1
        BCC     1$
        ERROR   MKTERR,-1,0,0
1$:
```

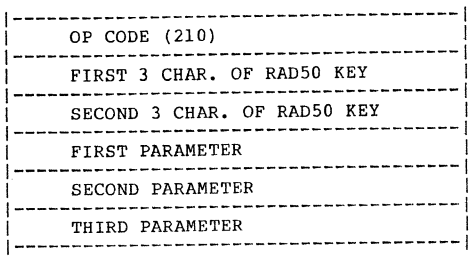
In this example you see an attempt to execute a mark time directive. If the directive fails, you want an error message and some attempt to recover from the error. The recovery may be as simple as aborting the task. The error macro definition follows:

```
        .MACRO  ERROR,P1,P2,P3,P4
;
; MACRO TO SET UP INSTRUCTION BLOCK FOR ERROR MESSAGES
;
;       ERROR   KEY,TYPE,PRAM1,PRAM2
;
```

```
;               KEY <= 6 RAD50 CHARACTERS
;               TYPE = -1 FATAL ERROR
;                    = -2 FOR DELAYED RECOVERY
;                    = -3 FOR JUMP RECOVERY
;                    = 0 FOR MESSAGE ONLY
;       SEE DESCRIPTION IN ERRDIS MODULE
;
        .WORD   210             ; ERROR HANDLING INSTRUCTION
        .NCHR   TEMPN,P1
        .IIF LT 6-TEMPN .ERROR ; RAD50 NAME > 6 CHARACTERS
        .IF  LT 3-TEMPN
        .RAD50  /P1/
        .IFF
        .RAD50  /P1/
        .RAD50  /   /
        .ENDC
        .WORD   P2,P3,P4
        .ENDM   ERROR

        .MACRO  SAVRG
        JSR     R5,$SAVRG
        .ENDM
```

The error control block looks like:



where:

1. The two word key is used to both identify the error, and as a key into an expanded message table. If a match is found in the message table, the message is printed in addition to the normal information.

2. The first parameter specifies the type of error recovery to perform. This parameter takes the following values:

   * 0 -- Message only. The task continues execution after the error control block. The second and third parameter contain data to be printed. If the second parameter is 0, R0 will be printed. If the third parameter is 0, R5 will be printed.

   * -1 -- Fatal error. The message is printed, and the erroring task is aborted. The second and third parameters are treated as in the message only case.

   * -2 -- Recoverable error. The message is printed and the task continues processing at the location specified in the second parameter. The first byte of the third parameter is the number of seconds to wait before continuing, and the last byte of the third parameter is the event flag to use while waiting. If the seconds are zero, it will default to 1, if the event flag is zero, it will default to 1.

   * -3 -- Recoverable error. The message is printed and the task continues processing at the location specified in the second parameter. If parameter 3 is 0, R5 will be printed.

   * If some other value than the ones listed above, the error is printed, and the offending task is aborted.

For example, you would use -2 for an error that required operator intervention and -3 for an error exit from the routine. The error printout has the following form:

```
'TSKNAM' PS = XXXXXX, PC = XXXXXX, DSW = XXXXXX
ERROR 'KEYNAM', PRAMS = XXXXXX XXXXXX
```

Where the XXXXXXs are octal numbers for the values of the erroring task. KEYNAM is the keyword for the error and TSKNAM is the name of the task where the error occurred. If a message is associated with the error, the message line is printed before the standard display. Again, this could be easily modified for your special needs.

1.0 DESCRIPTION OF THE CODE

This section provides an overview of the operation of the error logging task. The basic concept of the error logger is to allow any active task to display an error without extra code in the task. The error task is invoked by the execution of a PDP-11 reserved instruction. The error logging task intercepts the system illegal instruction processing and checks if the instruction is the one specific to the error logging system. If the instruction is not the error logging instruction, the instruction is handled in the normal manner, otherwise, the error logger is invoked.

To install the illegal instruction intercept, the error task does a Connect-To-Interrupt directive to an unused vector. At this time, some PIC code is inserted into the system pool, and the illegal instruction vector is pointed to the new code. This code examines the illegal instruction and if it is the instruction specific to the error logger, will cause a transfer to the error logger via the vector specified by the connect to interrupt. The code in the error logging task is simplified as the task is built as /PR:5.

Information about the erroring task is gathered and a call to $FORK2 transfers the task from interrupt level to system state. Here, information about the error is placed in an AST block which is sent to the error logging task to be processed.

1.1 Initialization And Self Test

The following is the initialization and self test sections of the error logging task. The task initializes itself and then suspends. If the task is resumed, it generates a sequence of test error messages.

```
CINT:   CINT$   ERRVEC,120000,ERRINT,ERRINI,PR7,ASTAD

START:
        DIR$    #CINT           ; CONNECT TO THE VECTOR,
        BCC     10$             ; AND INIT IT, IF ERROR
        IOT                     ; JUST CRASH

10$:    SPND$S                  ; ELSE SUSPEND FOREVER
;
; IF THIS TASK IS RESUMED, IT WILL SELF TEST
;
        ERROR   AAAAA,0,1,2     ; PRINT ERROR MESSAGE WITH
                                ;  DEFAULT PARAMETERS

        MOV     #3,R0
        MOV     #4,R5
        ERROR   BBBBB,0,0,0     ; PRINT ERROR MESSAGE
        BR      30$             ; NEXT MESSAGE SHOULD BE DDDDD

20$:    ERROR   CCCCC,-3,40$,6  ; NEXT MESSAGE WILL BE
                                ;    A DELAY MESSAGE

30$:    ERROR   DDDDD,-3,20$,5  ; CCCCC IS NEXT MESSAGE

40$:    ERROR   EEEEE,-2,50$,<<15*400>+5> ; CHECK FOR 5 SECOND DELAY

50$:    ERROR   FFFFF,-2,60$,<<0*400>+0>  ; CHECK DEFAULT DELAY
                                          ;   AND EVENT FLAG

60$:    ERROR   DONE,-3,10$,0   ; ALL DONE
```

Note that the interrupt priority is 7 and that the base address of the connect code is at the start of the task (120000).

1.2 AST Handler

The next section of code is the AST routine that displays the errors. This routine is easily modified for your special needs. One nice enhancement would be to send the erroring tasks TI terminal number to the AST routine so that the error would also be displayed on that terminal.

```
;
; INPUTS:
;       INFORMATION IS ON THE STACK, ALL REGISTERS MUST BE SAVED
;       THE INFORMATION  ON THE STACK IS:
;         2 WORDS OF RAD50 TASK NAME ISSUING THE INSTRUCTION
;         PS OF THAT TASK
;         PC OF THAT TASK
;         DSW OF THAT TASK
;         2 WORDS OF RAD50 ERROR KEYWORD
;         FIRST PARAMETER
;         SECOND PARAMETER
;
; EQUATED SYMBOLS
;
BEL = 7         ; ASCII CODE FOR BELL
        .ASECT
. = 0
                        ; STACK PARAMETERS

REGS:   .BLKW   6       ; R5-R0 R5 IS FIRST
ERTNM:  .BLKW   2       ; 2 WORDS OF RAD50 TASK NAME FROM ERROR
ETPS:   .BLKW   1       ; ERRORING TASKS PS
ETPC:   .BLKW   1       ; ERRORING TASKS PC
ETDSW:  .BLKW   1       ; ERRORING TASKS DSW
EKEY:   .BLKW   2       ; 2 RAD50 WORDS WITH KEY NAME
ETP1:   .BLKW   1       ; FIRST PARAMETER
ETP2:   .BLKW   1       ; SECOND PARAMETER

        .PSECT

        .ENABL  LSB

ASTAD:
        .IRPC   REG,<012345>    ; SAVE THE REGISTERS
        MOV     R'REG,-(SP)
        .ENDM
;
; GET THE INFORMATION FROM THE INTERRUPT PART AND DISPLAY IT
;
        MOV     SP,R0           ; GET THE ADDR OF THE ERROR DATA BLOCK
        MOV     EKEY(R0),R1     ; GET THE FIRST WORD OF THE KEY
        MOV     #MSGKYS,R2      ; AND THE ADDRESS OF THE KEYS
10$:    CMP     R1,(R2)         ; IS THE FIRST PART OF THE KEY THE SAME?
        BNE     20$
                                ; HERE THE FIRST PART OF THE KEY MATCHES
        CMP     EKEY+2(R0),2(R2) ; DOES THE SECOND PART MATCH?
        BNE     20$             ; IF NE NO, SO CONTINUE
```

```
        BR      50$

20$:    ADD     #6,R2           ; ADVANCE TO NEXT KEY
        TST     (R2)            ; CHECK NOT END OF KEYS
        BNE     10$             ; KEY NOT FOUND, => NO MESSAGE OTHER
                                ;   OTHER THEN KEY ITSELF
;
; HERE, CONVERT KEY TO ASCII, AND PRINT IT AND OTHER VARIABLES
;
        .PSECT  MESFMT,RO,D
        .NLIST  BEX

MG1:    .ASCII  <BEL><BEL>/'%2R' PS = %P, PC = %P, DSW = %P%N/
        .ASCIZ  /ERROR '%2R', PRAMS = %2P%N/
        .LIST   BEX
        .PSECT

30$:    MOV     SP,R2           ; SAVE POINTER TO PARAMETERS
        ADD     #ERTNM,R2
        SUB     #200.,SP        ; SAVE SPACE FOR LINE ON STACK
        MOV     SP,R0
        MOV     SP,R5
        MOV     #MG1,R1         ; POINT TO FORMAT INFORMATION
        CALL    $EDMSG          ; FORMAT IT
        CMP     #200.,R1        ; CHECK STACK DIDN'T GET WIPED
        BGE     40$
        IOT

40$:    QIOW$S  #IO.WBT,#TTLUN,#TTEV,,,,<R5,R1,#40>
        ADD     #200.,SP        ; CLEAN UP STACK
        BR      60$
;
; HERE PRINT MESSAGE, AND VARIABLES
;
50$:
        MOV     4(R2),R1        ; GET ADDRESS OF MESSAGE
        MOVB    (R1)+,R2        ; GET SIZE OF MESSAGE
        QIO$S   #IO.WBT,#TTLUN,,,,,<R1,R2,#40>
        BR      30$

60$:    .IRPC   REG,<543210>    ; RESTORE THE REGISTERS
        MOV     (SP)+,R'REG
        .ENDM

        ADD     #2*NUMPRM,SP    ; CLEAN OFF PARAMETERS
        ASTX$S

        .DSABL  LSB
```

There isn't much magical about this routine, it is just a normal AST receiver. Well, almost normal as the number of parameters on the stack isn't a standard number, but if you read the last article you will not be surprised by that.

1.3 Intercepting The Illegal Instruction Trap

The next routine is called when the connect to interrupt directive is executed. This routine is responsible for inserting some code into the system pool space and intercepting the illegal instruction trap. The format of the intercept code in the pool follows the form that was described by a paper in the DECUS proceedings<1>. The error logger could just as well be inserted into the system with the ISF task as described in that paper. However this task was developed for a stand alone RSX-11S task and needed to insert the code itself.

```
;+
;  **-ERRINI-CONNECT TO INTERRUPT ENABLE/DISABLE ROUTINE
;
; THIS ROUTINE ENABLES/DISABLES THE ILLEGAL INSTRUCTION INTERCEPT
; IF CARRY IS CLEAR, IT ENABLES, IF SET, IT DISABLES
; THIS MODULE ALLOCATES SOME POOL MEMORY, PLACES THE ILLEGAL INSTRUCTION
; INTERCEPT CODE INTO THE POOL AREA, MODIFIES XDT TO NOT PLAY WITH THE
; ILLEGAL INSTRUCTION TRAP VECTOR (ONLY IF XDT IS PRESENT) AND RETURNS.
;
; THE DISABLE OF THIS OPERATION UNDOES ALL OF THE ABOVE
;
; INPUTS:
;       CARRY SET TO DETACH FROM VECTOR, CLEAR TO ATTACH TO VECTOR
;
; OUTPUTS:
;       NONE
;
; SIDE EFFECTS:
;
;    MODIFIED EXTERNALS
;
;    OTHER SIDE EFFECTS
;       POLADD,POLSIZ ARE SET TO THE ADDRESS OF THE POOL AREA, AND ITS SIZE
;       THE ILLEGAL INSTRUCTION TRAP VECTOR IS INTERCEPTED.
;
; REGISTER USAGE:
;       R1,R4,R5 ARE SAVED
;
;-
ERRINI:
        BCC     10$             ;; CONNECT TO ILLEGAL INSTRUCTION VEC
        JMP     ABOPRO          ;; IF CARRY SET, DISCONNECT

10$:    MOV     R5,-(SP)        ;; SAVE REGISTERS
        MOV     R4,-(SP)
        MOV     R1,-(SP)
```

<1> Extending RSX-11M by Use of the Illegal Instruction Trap DECUS PROCEEDINGS Vol 6, No. 2, Pg. 467. This paper describes a means of inserting code into the system pool. It is good background reading for parts of this article

```
        MOV     #EPOLCD-POOLCD,R1 ;; NEED TO ALLOCATE POOL SPACE TO PUT
        CALL    $ALOCB            ;; TRAP INTERCEPT CODE
        BCC     20$
        IOT                       ;; IF NO POOL SPACE, CRASH SYSTEM

20$:    MOV     R0,POLADD         ;; SAVE ADDRESS OF ALLOCATED BLOCK
        MOV     R1,POLSIZ         ;; AND SIZE OF ALLOCATED BLOCK
        MOV     $TKTCB,MYTCBA     ;; AND WHILE AT IT, SET MY TCB ADDRESS

        MOV     #<EPOLCD-POOLCD>/2,R2   ;; GET THE NUMBER OF BYTES TO COPY
        MOV     #POOLCD,R3        ;; AND THE ADDRESS OF CODE TO MOVE
                                  ;; TOO THE POOL

30$:    MOV     (R3)+,(R0)+       ;; DO THE MOVE
        SOB     R2,30$            ;; UNTIL FULLY COPIED

        MOV     POLADD,R0         ;; GET THE START ADDRESS OF POOL CODE
        ADD     #2,R0             ;; BUMP PAST THE "ILL"
        MOV     POLSIZ,(R0)+      ;; SAVE THE NUMBER OF BYTES IN CODE
        MOV     @#ILLVEC,(R0)+    ;; SAVE THE ORIGINAL VECTOR
        MOV     #NOP,(R0)         ;; TURN ON THE CODE

        MOV     #$XDT,R2          ;; CHECK IF XDT IS AVAILABLE
        BEQ     40$               ;; IF EQ NO

        SUB     #XDTFIX,R2        ;; CHECK IF XDT IS CONSISTENT
        CMP     @#ILLVEC,(R2)
        BNE     40$
        MOV     R0,(R2)           ;; ADJUST XDT

40$:    MOV     R0,@#ILLVEC       ;; POINT ILLEGAL INSTRUCTION TRAP TO
                                  ;; THIS NEW CODE
        MOV     (SP)+,R1          ;; RESTORE REGISTERS
        MOV     (SP)+,R4
        MOV     (SP)+,R5
        RETURN                    ;; AND RETURN TO TASK
```

This code inserts the PIC code that starts at POOLCD into the system pool. It modifies XDT to allow executive debugging with this code inserted<2>. The code that is inserted into the system pool examines the instruction that caused the illegal instruction trap. If the instruction is recognised by the error system, this code simulates an interrupt via the unused interrupt vector that the error task connected to. Otherwise, the illegal instruction is passed to the original illegal instruction trap handler.

```
;
; CODE THAT GETS LOADED INTO THE POOL
```

<2> XDT resets the illegal instruction vector whenever it proceeds from a breakpoint. Thus resetting our intercept. The modification causes XDT to bypass resetting the illegal instruction vector.


```
;  >>>>>> THIS CODE MUST BE PIC <<<<<
;
POOLCD:
        .RAD50  /ILL/             ;;; SHOW THIS CODE IS ATTACHED TO THE
        .WORD   0,0               ;;; ILLEGAL INSTRUCTION VECTOR
        RTI                       ;;; INITIALLY, TURN OFF CODE

EP:     MOV     R5,-(SP)          ;;; SAVE R5
        MOV     2(SP),R5          ;;; AND GET TASK PC FROM STACK
        MFPI    -(R5)             ;;; GET THE INSTRUCTION THAT CAUSED THE TRAP
        CMP     (SP),#210         ;;; IS IT OUR INSTRUCTION?
        BEQ     100$

        ADD     #2,SP             ;;; NO! SO CLEAR THE STACK
        MOV     (SP)+,R5          ;;; RESTORE R5
        JMP     @EP-4             ;;; AND PROCESS NORMALLY
;
; HERE IT IS OUR ILLEGAL INSTRUCTION
;
100$:   ADD     #2,SP             ;;; CLEAN THE STACK
        MOV     (SP)+,R5          ;;; RESTORE R5
        MOV     @#ERRVEC,-(SP)    ;;; AND JUMP VIA THE VECTOR WE ARE CONNECTED TO
        JMP     @(SP)+            ;;; POSITION INDEPENDENT INDIRECT JUMP
EPOLCD:                           ;;; END OF THE CODE INSERTED INTO THE POOL
```

The illegal instruction trap vector points to EP-2. EP-4 contains the address of the original service routine. If the error logging task exits, the modifications to XDT must be removed along with the intercept code from the system pool.

```
ABOPRO:
        MOV     R1,-(SP)          ;; SAVE ITB POINTER
        MOV     @#ILLVEC,R0       ;; GET CURRENT VECTOR TO FIND OUR CODE
        MOV     -2(R0),@#ILLVEC   ;; RESET THE ILLEGAL INSTRUCTION VEC
        MOV     #$XDT,R2          ;; RESET XDT
        BEQ     1$

        SUB     #XDTFIX,R2
        MOV     -2(R0),(R2)
1$:     MOV     POLADD,R0         ;; GET THE ADDRESS OF THE POOL CODE
        MOV     POLSIZ,R1         ;; AND THE SIZE
        CALL    $DEACB
        MOV     (SP)+,R1          ;; RESTORE ITB POINTER
        RETURN
```

The next routine contains a large amount of magic. It copies the error parameter block from the user task and, if required, adjusts the issuing tasks program counter. After entering system state by a call to $FORK2, other actions on the issuing task are performed. These actions are aborting or delaying execution of the issuing task. Finally an AST is sent to the error logging task.

```
;+
;  **-ERRINT-PROCESSES ERROR INTERRUPTS
;
```

```
; THIS MODULE RECEIVES THE 210 INSTRUCTION CODE.
; IT SAVES THE ERROR BLOCK PARAMETERS, DOES ANY PROCESSING OF THE
; PARAMETERS (CHECKS WHAT TO DO WITH THE ISSUING TASKS PC).
; FORKS AND PLACES THE TASK IN THE APPROPRIATE STATE
;
; INPUTS:
;       R5 POINTS TO THE ITB
;       R4 IS SAVED AND MAY BE USED
;
; OUTPUTS:
;       NONE
;
; SIDE EFFECTS:
;
;    MODIFIED EXTERNALS
;
;    OTHER SIDE EFFECTS
;       MANY, TASK MAY BE ABORTED, MADE TO MARK TIME, TASKS PC MAY BE
;       MOVED AS DIRECTED BY THE PARAMETERS ETC.
;
; REGISTER USAGE:
;
;       R5 POINTS TO ITB
;       R4 IS SAVED AND MAY BE USED, ALL OTHER REGISTERS MUST BE SAVED AND
;       RESTORED
;
;-

;
; LOCAL DATA
;
ERL1:   .WORD   0
ERL2:   .WORD   0
PER1:   .WORD   0
PER2:   .WORD   0
PER3:   .WORD   0

ERRINT:                         ;;; HERE IF ILLEGAL INSTRUCTION
                                ;;; FIRST 2 WORDS ARE RAD50 REP
                                ;;;   OF NAME OF ERROR
        MOV     10(SP),R4       ;;; GET PC OF TASK
        MFPI    (R4)+           ;;; GET FIRST WORD OF RAD50 ERROR
        MOV     (SP)+,ERL1      ;;;
        MFPI    (R4)+           ;;;
        MOV     (SP)+,ERL2      ;;;
        MFPI    (R4)+           ;;;
        MOV     (SP)+,PER1      ;;;
        MFPI    (R4)+           ;;;
        MOV     (SP)+,PER2      ;;;
        MFPI    (R4)+           ;;;
        MOV     (SP)+,PER3      ;;;

        TST     PER1            ;;; CHECK IF MESSAGE ONLY
        BMI     20$             ;;; IF MI NO
```



```
        MOV     R4,10(SP)       ;;; SET NEW PC AS ONLY MESSAGE IS REQUIRED
10$:    CALL    @#$FORK2        ;;;
        CLR     (R3)            ;; CLEAR THE FORK BLOCK
        CALLR   SNDMSG          ;; AND SEND AST TO TASK STATE

20$:    CMP     #-1,PER1        ;;; CHECK IF TO ABORT TASK
        BNE     40$             ;;; NE NO

30$:    CALL    @#$FORK2        ;;;
        CLR     (R3)            ;; CLEAR THE FORK BLOCK
32$:    MOV     #S.CABO,R0      ;; ABORTING FROM ILLEGAL INSTRUCTION
        CALL    $ABCTK          ;; ABORT THE CURRENT TASK
        CALLR   SNDMSG          ;; AND SEND AST TO TASK STATE

40$:    CMP     #-2,PER1        ;;; CHECK IF OFFSET JMP
        BNE     50$

        MOV     PER2,10(SP)     ;;; ADJUST PC
        CALL    @#$FORK2        ;;;
        CLR     (R3)            ;; CLEAR FORK BLOCK
        MOV     #2,-(SP)        ;; SET THE UNITS TO SECONDS
        MOVB    PER3,-(SP)      ;; GET THE NUMBER OF SECONDS TO WAIT
        BNE     41$             ;; CHECK SPECIFIED
        INC     (SP)            ;; DEFAULT IS 1

41$:    CLRB    1(SP)           ;; MAX 255
        MOV     SP,R3           ;; POINT R3 TO BLOCK
        CALL    $CVRTM          ;; AND CONVERT THE TIME
        MOV     R0,2(SP)
        MOV     R1,(SP)
42$:    MOVB    PER3+1,R0       ;; GET THE EVENT FLAG TO USE
        BNE     43$             ;; EVENT FLAG OF 0 IMPLIES DEFAULT
        INCB    PER3+1
        BR      42$

43$:    MOV     $TKTCB,R5       ;; AND THE TCB OF THE CURRENT TASK
        CALL    $CEFI           ;; AND CONVERT TO MASK WORD AND ADDRESS
        BCC     45$
44$:    ADD     #4,SP           ;; CLEAN UP STACK
        BR      32$

45$:    CMPB    PER3,#64.       ;; CHECK IF GLOBAL EVENT FLAG
        BHI     44$             ;; IF SO GO TO 44 AND ABORT

        BIC     R0,(R1)         ;; CLEAR THE EVENT FLAG
        MOV     R0,-(SP)        ;; SAVE EVENT FLAG MASK WORD
        MOV     R1,-(SP)        ;; AND THE EVENT FLAG MASK ADDRESS
        MOV     T.PCB(R5),R4    ;; GENERATE POINTER TO TASK HEADER
        MOV     P.HDR(R4),R4
        INC     T.ST2(R5)       ;; PUT TASK IN WAIT FOR STATE
        MOV     2(SP),H.EFLM(R4) ;; SAVE WAIT FLAG
        MOV     (SP),H.EFLM+2(R4)
        CALL    $SETRT          ;; SET SCHEDULE REQUEST
        MOV     #C.MRKT,R4      ;; MAKE CLOCK BLOCK A MARK TIME BLOCK
        CALL    $ALCLK          ;; ALLOCATE A CLOCK BLOCK
```

```
        MOVB    PER3+1,C.EFN(R0) ;; SAVE MARK TIME EVENT FLAG
        CLR     C.AST(R0)        ;; CLEAR THE AST ADDRESS
        MOV     (SP)+,C.DST(R0)  ;; SET THE EVENT FLAG DESTINATION
        MOV     (SP)+,C.SRC(R0)  ;; AND THE VALUE TO BE SET
        MOV     (SP)+,R2         ;; GET THE TIME TO WAIT
        MOV     (SP)+,R1
        CALL    $CLINS           ;; AND INSERT THE MARK TIME BLOCK IN THE
        CALLR   SNDMSG           ;; CLOCK QUEUE, AND SEND AST TO TASK STATE


50$:    CMP     #-3,PER1         ;;; CHECK IF ABSOLUTE JMP
        BNE     30$              ;;;
        MOV     PER2,10(SP)      ;;;
        BR      10$              ;;;
```

The call to SNDMSG just sends an AST to the user mode part of the task. Starting at label 40$ is the code that causes the task to delay for a time. One caution here, this method can only be used with an executing task that is in memory. We are sure of that condition in this example, however other cases may not be as simple.

The next routine just sends the error information to the AST part of this task. The method has already been explained in the last multi tasker.

```
;+
;  **-SNDMSG-SEND MESSAGE (VIA AST)
;
; THIS MODULE SENDS AN AST BLOCK TO THIS TASK.
; THE AST PARAMETERS ARE DESCRIBED IN THE AST RECEIVER MODULE
;
; INPUTS:
;       ERL1-PER3 -- INFORMATION FROM THE AST
;
; OUTPUTS:
;       GENERATES AN AST BLOCK WHICH IS QUEUED TO THE TASK
;
; SIDE EFFECTS:
;
;    MODIFIED EXTERNALS
;
;    OTHER SIDE EFFECTS
;
; REGISTER USAGE:
;       ALL REGISTERS ARE FREE TO BE USED
;
;-

SNDMSG:
        MOV     #<5+NUMPRM>*2,R1 ;; ALLOCATE AN AST BLOCK FOR ERRLOG
        CALL    $ALOCB           ;; ALLOCATE SOME POOL
        BCC     10$
        IOT                      ;; CRASH SYSTEM

10$:    MOV     R0,R2            ;; COPY ADDRESS OF BLOCK
        MOV     R0,ASTADD        ;; SAVE ADDRESS OF BLOCK
```



```
        CLR     (R2)+            ;; CLEAR THE LINK
        MOV     R1,(R2)+         ;; SET IN SIZE OF BLOCK
        MOV     #<7+NUMPRM>*2,(R2)+ ;; SET NUMBER OF BYTES FOR STACK
        MOV     #ASTAD,(R2)+
        MOV     #NUMPRM,(R2)+    ;; SPECIFY THE NUMBER OF PARAMETERS
;
; LOAD STANDARD PARAMETERS
;
        MOV     $TKTCB,R0        ;; GET THE POINTER TO THE TCB
        MOV     T.NAM(R0),(R2)+  ;; GET THE TASK NAME
        MOV     T.NAM+2(R0),(R2)+
        MOV     22(SP),(R2)+     ;; GET THE TASK PS
        MOV     20(SP),(R2)+     ;; GET THE TASK NEW PC
        MFPI    @#$DSW           ;; GET THE USER DSW
        MOV     (SP)+,(R2)+
        MOV     ERL1,(R2)+       ;; SAVE ERROR LABEL
        MOV     ERL2,(R2)+

        TST     PER1             ;; CHECK FOR TYPE OF INFO TO TASK
        BNE     50$
;
; HERE FOR NORMAL MESSAGE NO TASK AFFECT
;
20$:    MOV     PER2,(R2)+       ;; GET THE FIRST PARAMETER
        BNE     30$              ;; IF DEFAULT USE

        MOV     4(SP),-2(R2)     ;; R0 AS PARAMETER
30$:    MOV     PER3,(R2)        ;; GET THE SECOND PARAMETER
        BNE     40$              ;; IF DEFAULT USE
        MOV     16(SP),(R2)      ;; R5 AS PARAMETER
40$:    BR      70$

50$:    CMP     PER1,#-1         ;; CHECK FOR FATAL ERROR
        BEQ     20$              ;;  TREAT AS NORMAL ERROR

        CMP     PER1,#-2         ;; CHECK FOR RECOVERABLE ERROR
        BNE     60$

        CLR     (R2)+            ;; IF RECOVERABLE ERROR, NO PARAMETER
        CLR     (R2)
        BR      70$

60$:    CMP     PER1,#-3         ;; CHECK FOR RECOVERABLE ERROR
        BNE     20$
        CLR     (R2)+            ;; IF RECOVERABLE, NO PARAMETER 1
        BR      30$

70$:    MOV     MYTCBA,R0        ;; GET TASK ERRDIS TASK TCB
        MOV     ASTADD,R1        ;; GET AST BLOCK ADDRESS
        CALLR   @#$QASTT         ;; AND PLACE ON AST QUEUE
```

All that is left is to see how the table of extra error information lines is generated. However, this is only the method I used for the RSX-11S system. A unified error file is quite possible and is recommended.

```
;+
; FILE DESCRIPTION
;
;       THIS FILE CONTAINS ERROR MESSAGES
;
;       THE MESSAGES ARE STORED UNSORTED IN THE FOLLOWING MANNER:
;
;               PSECT MSGKEY CONTAINS THE FOLLOWING STRUCTURE
;                       FIRST 3 CHARACTERS OF KEY
;                       SECOND 3 CHARACTERS OF KEY
;                       POINTER TO MESSAGE IN PSECT MSGMSG
;
;               THE KEYS ARE TERMINATED BY A KEY WITH SPACES AS THE
;               FIRST 3 CHARACTERS = IE 0
;
;               PSECT MSGMSG CONTAINS THE FOLLOWING STRUCTURE
;                       MESSAGE SIZE, MESSAGE
;
;-
;
        .MACRO  MAKMSG KEY,MES
        .PSECT  MSGMSG
        .NCHR   TEMSZ,<MES>
        TEMP = .
        .BYTE   TEMSZ
        .ASCII  /MES/
        .PSECT  MSGKEY
        .RAD50  /KEY/
        .WORD   TEMP
        .PSECT

        .ENDM   MAKMSG

        .PSECT  MSGKEY,RO,D
MSGKYS::
        .PSECT  MSGMSG,RO,D
MSG::
        .PSECT

        MAKMSG  AAAAA,<TEST MESSAGE>
        MAKMSG  <       >,<DONE>

        .PSECT  MSGMSG
        .EVEN

        .PSECT  MSGKEY
        .EVEN

        .PSECT


        .END
```


LIBRARY FILE COMPRESSION

Jim Palmer

Science Applications, Inc.
Irvine, California

At Science Applications, program development is eased by the use of a number of command file utilties. One command file, COMPRESS.CMD, enables a user to compress any type of library file and then deallocate unneeded disk blocks. The utility is especially useful in resource critical, development intensive environments.

```
;       COMPRESS.CMD
.;
.; REVISION 4, 23-NOV-81
.; REVISION 3, 16-NOV-81
.; REVISION 2, 04-NOV-81
.; REVISION 1, 31-AUG-81
.; REVISION 0, 20-APR-81
.; PROGRAMMER: JIM PALMER
.;
.; THIS IS A COMMAND FILE TO:
.;      COMPRESS TARGET LIBRARY
.;      DEALLOCATE UNEEDED DISK BLOCKS
.;
.; TO INVOKE:
.;      TYPE AT MCR LEVEL
.;
.;        @COMPRESS LIBNAME.EXT
.;
.;          WHERE
.;            LIBNAME - TARGET LIBRARY FILE
.;            EXT - LIBRARY EXTENSION
.;        IF .EXT OMITTED
.;          THEN
.;            .EXT DEFAULTS TO .OLB
.;
.ENABLE SUBSTITUTION
.ENABLE QUIET
.;
.; PARSE @ COMMAN EXPRESSION
.;
.PARSE COMMAN " .;" CMDFLE F E ICATCH
.;
.;              ! DESCRIPTOR OF SUBEXPRESSIONS
.;              ! CMDFLE - CONTAINS THIS COMMAND FILE NAME [NOT USED]
.;              ! F - LIBRARY FILE NAME
.;              ! E - LIBRARY FILE EXTENSION
.;              ! ICATCH - CATCH ALL SYMBOL [EXTRA PARAMETERS - NOT USED]
.;
```

```
.;      IF FILE NAME EMPTY [NULL]
.; OR IF FILE EXTENSION EMPTY [NULL]
.;      THEN
.;        FORCE DEFAULT(S)
.;
.IF F = "" .SETS F "OLDLIBARY"
.IF E = "" .SETS E "OLB"
.;
.; TEST FOR EXISTENCE OF LIBRARY
.TESTFILE 'F'.'E'
.;
.; IF LIBRARY DOES NOT EXIST THEN SEND MESSAGE TO TI: AND EXIT
.IF <FILERR> = 230. .DISABLE QUIET
.IF <FILERR> = 230. ; Library 'F'.'E' not found  ! FCS error -26
.IF <FILERR> = 230. .STOP
.;
.; ELSE RETURN FCS ERROR CODE
.SETN FILERR <FILERR>
.IF <FILERR> <> 1 .DISABLE QUIET
.IF <FILERR> <> 1 ; FCS error 'FILERR'
.IF <FILERR> <> 1 .STOP
.;
.; COMPRESS LIBRARY
.XQT LBR 'F'.'E'/CO='F'.'E'
.;
.; WAIT FOR LBR COMPLETION
.WAIT LBR
.;
.; TEST LBR EXIT STATUS
.SETN EXSTAT <EXSTAT>
.IF <EXSTAT> <> 1 .DISABLE QUIET
.IF <EXSTAT> <> 1 ; LBR exit status error 'EXSTAT'
.IF <EXSTAT> <> 1 .STOP
.;
.; CREATE PIP /LI DIRECTORY TEMPORARY FILE
.;   THIS FILE IS SUBSEQUENTLY READ TO DETERMINE
.;   NUMBER OF DISK BLOCKS USED BY NEWLY COMPRESSED LIBRARY.
.;
.XQT PIP 'F'.LLI='F'.'E'/LI
.;
.; WAIT FOR PIP COMPLETION
.WAIT PIP
.;
.; TEST PIP EXIT STATUS
.SETN EXSTAT <EXSTAT>
.IF <EXSTAT> <> 1 .DISABLE QUIET
.IF <EXSTAT> <> 1 ; PIP exit status error 'EXSTAT'
.IF <EXSTAT> <> 1 .STOP
.;
.; OPEN .OLB DIRECTORY TEMPORARY FILE
.OPENR #0 'F'.LLI
.;
.; TEST FCS ERROR STATUS
.SETN FILERR <FILERR>
.IF <FILERR> <> 1 .DISABLE QUIET
```



```
.IF <FILERR> <> 1 ; FCS error 'FILERR'
.IF <FILERR> <> 1 .STOP
.;
.; INITIALIZE RECORD COUNT
.SETN RECKNT 0
.;
.; CONTINUE
.10:
.;
.; READ 1 UNIT RECORD
.READ #0 INBUF
.;
.; TEST FOR END OF FILE
.IFT <EOF> .GOTO EOFERR
.;
.; INCREMENT RECORD COUNTER
.INC RECKNT
.;
.; TEST CONTENTS OF RECKNT
.;   IF .NE. RECORD NUMBER 3
.;     THEN
.;       BRANCH BACK AND READ NEW RECORD
.;
.IF RECKNT <> 3 .GOTO 10
.;
.; RECORD NUMBER 3 ASSUMED FOUND
.;   CLOSE PIP /LI DIRECTORY TEMPORARY FILE
.;   DELETE PIP /LI DIRECTORY TEMPORARY FILE
.;
.CLOSE #0
PIP 'F'.LLI;*/DE/NM
.;
.SETS B "0"
.;
.PARSE INBUF "." B ICATCH
.;
.;              ! DESCRIPTION OF SUBEXPRESSIONS
.;              ! B - NUMBER OF DISK BLOCKS USED
.;              ! ICATCH - CATCH ALL SYMBOL [EXTRA PARAMETERS - NOT USED]
.;
.; COPY LIBRARY AND PREALLOCATE DISK BLOCKS
PIP 'F'.'E'/PU
.;
.; PROCESSING COMPLETED
.; TERMINATE PROGRAM
.;
.STOP
.;
.; RECORD NUMBER 3 ASSUMED NOT FOUND [ERROR]
.;
.EOFERR:
.;
.DISABLE QUIET
; File 'F'.LLI Record number 3 not found
.STOP
```

* ORTHO performs a least-squares fit.
* PCS interpolates a set of points using a piecewise-cubic spline.
* SFT does a Fourier transform on a set of points.
* WORTHO performs a weighted least-squares fit.

Sorting

* SORTIT does a bubble sort.

String Parsing

* ITWAS finds the character position of a given character in a string.

Terminal I/O

* TOUTIN does a read-after-prompt.

- o SETS is a utility that will display all the characters in a particular set.
- o TVG is a view graph maker with very extensive user controls.

1980 - Fall - San Diego

[370,040] COMMAND

COMMAND is a package that provides a Fortran interface to the TPARS string parser. It includes a set of TECO macros that generate a Fortran common block that defines the state tables and the necessary run-time code.

1980 - Fall - San Diego

[370,041] COMMAND FILES

This account has various command files for a variety of utility functions.

- o DIRLST gives a compressed directory from a PIP [*,*]/TB.
- o LOGIN allows login to a UIC group.
- o MEMO creates runoff memos.
- o PIPTR searches for and truncates files with non-allocated blocks.

1980 - Fall - San Diego

[370,050] FISMU

FISMU is a RSX-11M task to emulate FIS instructions. It creates a dynamic region with the emulation code and patches the illegal instruction trap to pass FIS instructions to this region.

1980 - Fall - San Diego

[370,060] NOS SUBROUTINES

This account has four Fortran callable subroutines:

- o ASCEBC is a routine to convert ASCII to EBCDIC.
- o CDATE is a routine to convert a Julian date to the corresponding calendar date.
- o EBCASC is a routine to convert EBCDIC to ASCII.
- o JDATE is a routine to convert a calendar date to the corresponding Julian date.

1980 - Fall - San Diego

[370,070] F4OTS

F4OTS is a very complete document describing the Fortran IV (FOR) OTS system. It is current for the latest release of FOR.

1980 - Fall - San Diego

[370,100] MAEACT

This account has an accounting system for IAS. The system is oriented towards accounting on an "project" basis and not by UIC.

1980 - Fall - San Diego

[370,110] VTECO

VTECO is a crt-style version of TECO. The distribution supports various terminals such as the VT52, VT100, ADM3A, and ADM42. The

editor is designed for user-implementation of other terminal types.

1980 - Fall - San Diego

[370,120] LOG

LOG is a set of RSX-11M utilities for providing charge-back accounting for RSX-11M systems. The implementation uses a statistical sample to measure task execution time, device usage, and terminal logon time.

1980 - Fall - San Diego

[370,130] INDEX

INDEX is a FOR or F4P cross reference utility.

1980 - Fall - San Diego

[370,140] MGT

MGT is a magtape utility that performs magtape copies to or from disk or magtapes. It does no data translation so it can be used to copy any format tape. It also hase basic tape manipulation commands.

1981 - Spring - Miami

[370,150] HEX

HEX is a utility to manipulate INTEL ASCII Hex-formatted files that are produced by many cross assemblers or linkers for 8-bit microprocessors.

1981 - Spring - Miami

[370,150] CLUNK

CLUNK is a subroutine callable from Macro, Fortran or Basic-Plus-2 that converts Datatrieve clunks to system date and time. It can also convert in the reverse direction.

1981 - Spring - Miami
1980 - Fall - San Diego

[371,004] BATTELLE SUBROUTINES

This account has the subroutine library used at BATTELLE. Included in the account are routines for device and file management, string manipulation, date/time functions, subtasking control, and miscellaneous routines. Included in the submission are the following modules:

Command Parsing

* GETCMD gets the PDS command line.

Conversion

* ACNVT converts an integer to ASCII.
* FCNVT converts a ASCII string to a floating number
* ICNVT converts a ASCII string to a binary number.
* CVTUIC converts an ASCII UIC to binary.

Data Manipulation

* MOVEB moves arrays or variables byte-by-byte.
* SWAPB swaps variables or arrays byte-by-byte.
* SWAPW swaps variables or arrays word-by-word.

Date/Time

* ADOW returns the ASCII day-of-the-week.
* AMON returns the ASCII month.
* CVTDAT converts ASCII date to binary and vice-versa.
* CVTTIM converts ASCII time to binary and vice-versa.
* DAYTIM returns the current date, time, and day-of-the-week.
* GETTIM returns the system time parameters.
* HAPO converts Gregorian to HAPO and vice-versa.
* JULIAN converts Gregorian to Julian and vice-versa.
* WKDAY returns the day-of-the-week for a specific day.

File I/O

* ATTACH attaches a device
* DETACH detaches a device
* DELETE deletes a file by name.
* FILNAM generates a filename string.
* GETFDB gets the Fortran FDB address.
* GETSIZ gets the number of blocks allocated to a file.
* GETUIC gets the current UIC.
* GETVER gets the open file's version number.
* NOLOCK opens a file with no lock bit.
* PROTEC sets the default file protection and ownership.
* RENAME renames a file.
* REOPEN reopens a file using the Fortran FDB.

Fortran OTS

* CHKPRV checks if task is running from a privilege account.
* EXITWS exits with status.
* GETUSR gets the username who is logged into the terminal.

String Parsing

* COMPAR compares two ASCII strings.
* COMPRS compress an ASCII string.
* CONCAT concatenates two ASCII strings.
* FILL pads a string with trailing spaces.
* LENGTH returns the length of a string.
* LFJUST left justifies a string.
* LOCASE converts upper case characters in a string to lower case.
* LOCATE finds the position in a string of a substring.
* RTJUST right justifies a string.
* STRIP removes trailing spaces.
* UPCASE converts lower case characters in a string to upper case.

Task Control

* SUBRTT spawns a real-time subtask.
* SUBTSK spawns a timesharing subtask.

Terminal I/O

* PROMPT issues a read with prompt QIO.
* RDTTY issues a read QIO to the terminal.
* WRTTY issues a write QIO to the terminal.

1980 - Spring - Chicago

[371,200] DALLOC

DALLOC is a IAS disk management/quota package.

1980 - Spring - Chicago

[372,004] BPR
[372,004] SAMSTAT

This account has two packages. BPR is a preprocessor for BASIC-PLUS-2 that allows program to be written without worring about line numbers. SAMSTAT is a statistics utility with many features.

1980 - Spring - Chicago

[373,001] TAP

TAP is a "universal magtape reader" with features to allow you to determine what format the tape is in and read the tape contents to disk.

NOTE

The [373,*] accounts have many programs submitted elsewhere in the SIG tapes. However, these versions have typically been modified to correct minor bugs and add on-line help facilities. Also, each submission in these accounts has documentation, not always present with the original submission. This makes these versions very nice and a definite consideration for deciding which version of an utility to use.

1981 - Spring - Miami

[373,002] FRG

FRG is a utility to output a histogram of the available free space sizes.

1981 - Spring - Miami

[373,003] FRC

FRC is a utility that forces MCR commands to another terminal. This is a modified version that uses the spawn directive so the task need not be mapped to the executive.

1981 - Spring - Miami

[373,0004] SRD

SRD is a directory utility with a wide range of file selection options. This version appears to be a superset of some of the other versions of SRD on the tape. Inspection of the documentation shows this version to have an extensive array of switches and options.

1981 - Spring - Miami

[373,005] FORMAT
[373,005] RTR

RTR is a program to read files from a RSTS/E disks. FORMAT is a program which takes the output from RTR and converts it to suitable input to the BP2 compiler.

1981 - Spring - Miami

[373,006] WTR

WTR converts a document created by WORD-11 into standard RUNOFF format.

1981 - Spring - Miami

[373,0007] FSV

FSV is a simplistic program to reover a deleted file.

1981 - Spring - Miami

[373,010] SND

SND is a tets program that can produce generalized sends to tasks and also act as a listener. This is useful for debugging tasks which use send/recieves to communicate or to send once-only style messages.

1981 - Spring - Miami

[373,011]

PIN is a procedure interpreter, i.e. a command file processor like IND without the special features. This is a new version of Jim Downward's original version that has been made even smaller.

1981 - Spring - Miami

[373,012] XRF

XRF is a Fortran cross referencing program.

1981 - Spring - Miami

[373,013] CCL

This is an alternate version of CCL that has had some code-cleanup and new features of recursion and dynamic task installation added.

1981 - Spring - Miami

[373,014] TCF

TCF is a simple program to output control characters to a terminal. It accepts octal numbers and outputs the characters back.

1981 - Spring - Miami

[373,015] FIN

FIN is a version of GREP. It performs pattern searches on files, reporting all occurences of a given string.

1981 - Spring - Miami

[373,016] CVL

CVL is a utility program to allow fields in the volume label to be changed. This version corrects a few bugs, is not mapped to the executive, and includes a help facility.

1981 - Spring - Miami

[373,017] DSRDMP

DSRDMP is a utility that snapshots the executive pool and then produces a dump of the structures found in it. It seems to be able to handle almost all of the standard RSX-11M V3.2 data structures.

1981 - Spring - Miami

[373,020]

HOL is a utility that will list out all free space on a disk (or holes larger than a given size). It will give the starting logical block number and size in decimal.

1981 - Spring - Miami

[373,021] VCO

VCO will compare two files block by block. It is primaryly intended for checking task images or object modules. It allows wild-cards in its input.

1981 - Spring - Miami

[373,022] CHK

CHK calculates block checksums for files (using the addition of all words).

1981 - Spring - Miami

[373,023] DOB

DOB is a Macro-11 object module disassmbler. This is an updated version of the [300,23] version and includes more instruction encoding, output in a form suitable to the assembler, and a built-in help feature.

1981 - Spring - Miami

[373,024] STB

STB is a simple program to interogate RSX11M.STB files to find out the address of a particular symbol.

1981 - Spring - Miami

[373,025] ICLDMP

This is some sort of magtape dump utility.

1981 - Spring - Miami

[373,026] MK3DEM

This appears to be some demonstration software for the submission in [373,102]

1981 - Spring - Miami

[373,027]

This is a program to invoke PMD to cause a snapshot dump of another task.

1981 - Spring - Miami

[373,030] TPC

TPC is a utility to copy a tape to a disk file and then make copies from the disk file. It is optimized for speed. This submission includes the only documentation ever produced for TPC.

1981 - Spring - Miami

[373,031] TAPBCK

This program appears to read a disk file and write EBCDIC records to tape.

1981 - Spring - Miami

[373,032] REW

REW is a simple program that mearly rewinds a tape (however, that is difficult to do without this program).

1981 - Spring - Miami

[373,033] TECO

This account has a version of TECO V35, together with extensive documention and general purpose TECO macros.

1981 - Spring - Miami

[373,034] BATCH

BATCH is a batch system for RSX-11M. This is a modified version of the KMS Fusion submission that is modified for support of input files as well as data files and errors corrected.

1981 - Spring - Miami

[373,035] SOS

This account had a version of the SOS editor from the RSX-11M Plus kits. However it was deleted because SOS is proprietary software of Digital. However, anyone with RSX-11M Plus can try using SOS on their RSX-11M system, it seems to work.

1981 - Spring - Miami

[373,101] USRLIB
[373,101] USRMAC

The account has a subroutine and macro library with many useful functions. Includinged in the subroutine library are the following entries:

Conversions

* ASCBIN converts ASCII to binary.
* ASCEBD converts ASCII to EBCDIC.
* ASCFNM converts an ASCII filename to a filename block.
* ASCRAD converts ASCII to RAD50.
* BINASC converts binary to ASCII.
* CNVTUC converts a string to all upper case.
* EBCASC converts EBCDIC to ASCII.
* FNMASC converts a filename block to a ASCII string.
* RADASC converts RAD50 to ASCII.

Date/Time

* CDATxy converts a date from type x to type y.
* CTIMxy converts a time from type x to type y.
* GTDATx gets todays date in type x.
* GTTIMx gets current time in type x.

The possible valid types for dates include the following:

I	Day, month, year as 3 byte values
A	ASCII notation (DD-MMM-YY)
D	Integer*2 days since January 1, 1950
E	Integer*2 encoded date (yyyyyyymmmmddddd)
W	Weekday number (0 = Sunday)

The possible valid types for times include the following:

I	Hours, minutes, seconds, and tenths of seconds
A	ASCII notation (HH:MM:SS.S)
D	Integer*4 seconds since midnight

String Parsing

* STDELE deletes a substring from a string.
* STEXCH exchanges characters in a string.
* STFILL initializes a string with a character.
* STFIND finds a substring in a string.
* STINSE inserts a substring in a string.
* STJUST right or left justifies a string.
* STLOCA locates one of a set of characters in a string.
* STRCMP compares two strings.
* STREPL replaces occurances of one character with another.

The library also has special routines for sorting and random number generation. The macro library has macros for setting up Fortran subroutine calls and stack operations.

1981 - Spring - Miami

[373,102] SCREEN

SCREEN is a collection of subroutines that provide terminal-type independent CRT handling. A wide-variety of entries are provided and the package supports use-defined definition of new terminal types.

1981 - Spring - Miami

[373,104] USRLIB

This account has the sources for the date/time routines found in USRLIB in [373,101].

1981 - Spring - Miami

[373,105] USRLIB

This account has the sources for the string routines documented in [373,101].

1981 - Spring - Miami

[373,107] TPARS DOCUMENTATION

This account has documentation on the use of TPARS. TPARS is a finite-state, tabel-driven parser supplied with your distribution kits.

1981 - Spring - Miami

[373,301] SUBMISSION STANDARDS

This account has documentation on the standards used for the software in the [373,*] accounts. This includes source file format, help files, command files, and denvelopment philosphy.

NOTE

These is neat!! If everyone in the future followed these ideas, my job in keeping this document would be much simplier.
Ralph Stamerjohn

1981 - Spring - Miami

[373,302] HELP FILES

This is probably the world's largest collection of help files for RSX-11M.

1981 - Spring - Miami

[373,310] RSX DOCUMENTATION

This account has various documentation files, including the following:

- o INTERAST is a paper on various techniques for intertask communications.
- o RMSINT is documentation on the internals of RMS.
- o RSXMOD is a note on how to modify the executive to support foriegn disk drives.
- o RSXPLU is a note on how to get RSX-11M Plus up and running on a system with foreign disk drives.
- o STNLON is a note on how to permantantly change the magtape vectors and CSR's in standalone versions of BAD, DSC, and PRESERVE
- o XDTHNT is a note on clever ways to use XDT for device drivers.

1981 - Spring - Miami

[374,002] VUE

VUE is a modified version of TECO V28 that provides screen editing capabilities without using TECO macros. It works with VT-100's and provides a window into the text while normal TECO commands are entered at the bottom.

1980 - Spring - Chicago

[374,300] BRU PATCH
[374,300] "C" COMPILER
[374,300] "C" ASSEMBLER
[374,300] ICE

This account has three submissions:

- o BRU PATCH is a patch to BRU to fix tape I/O to foreign tape drives.
- o "C" COMPILER and ASSMBLER are versions to "wet your appetite". Users are referred to the Structure Languages SIG Tapes.
- o ICE is an interactive, screen-mode editor for VT-series terminals.

1980 - Fall - San Diego

[375,001] 3M SUBMISSIONS
[375,001] DSKPAT

This account has general information about the [375,*] accounts. It also has a disk patching utility (DSKPAT) that knows about file header checksums and can read or wite both logical and virtual blocks.

1980 - Spring - Chicago
1980 - Fall - San Diego

[375,002] TCR

TCR is a RSX-11M or M-PLUS MCR command that outputs the current terminal status and characteristics.

1980 - Fall - San Diego

[375,003] TCU

TCU is a task to set the time of a TCU-100 or TCU-150 clock and reset the system time from the current TCU time.

1980 - Fall - San Diego

[375,004] CDC

CDC is a task to use the hardware formatting abilities of the Emulax/CDC 9766 disk drive to format packs.

1980 - Fall - San Diego

[376,001] DECUS LIBRARY

This account has various indexes to the DECUS library.

1980 - Spring - Chicago